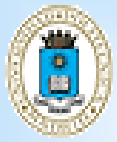


POLITÉCNICA

"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Facultad de Informática

TRABAJO FIN DE GRADO

DESARROLLO DE UNA APLICACIÓN MÓVIL ORIENTADA A M-LEARNING E IMPLEMENTACIÓN DE UNA CAPA DE SERVICIOS WEB PARA EL ACCESO REMOTO AL AFRICA BUILD PORTAL

Autor: Jose María Ramírez Barambones
Tutor: Víctor Manuel Maojo García
Cotutor: Máximo Ramírez Robles

MADRID, JUNIO DE 2014

A mis amigos, que me dan la vida, y a Cris,
la persona que más quiero en este mundo.

Me lo contaron y lo olvidé; lo vi y
lo entendí; lo hice y lo aprendí.

Confucio (551 AC-478 AC)

Agradecimientos

Que puedo decir. Podría empezar a nombrar personas y terminaría ocupando más páginas que el proyecto. Sin embargo, quiero destacar al menos a los más cercanos, los cuales sin ellos no hubiese podido realizar este trabajo ni estas palabras ni llegar a donde he llegado.

Primero quiero mencionar a mi hermano Alex y a mi madre María por ayudarme a tomar el camino correcto e insistir aun cuando a veces me olvido en escucharla, un auténtico ejemplo de superación. Mi abuela Ana con la cual el apoyo no entiende de distancias. Y en un lugar mejor mis dos abuelos y mi tío que desafortunadamente no volveré a ver pero sé que están conmigo y poder ser digno de su reconocimiento. No quiero olvidarme de mi tío Maxi que es como un padre para mí y que tras tantos altibajos, juntos hemos podido flanquear barreras imposibles.

En segundo lugar quiero mencionar al resto de mi familia que componen mis amigos, los cuales no puedo evitar llamarlos hermanos. Mario, Vanesa y Chelo, esos Berzal testarudos, con los que da gusto darse de cabezazos; la experiencia personificada. Una elite a la que pertenezco con orgullo, "Los hombres de Coocker": Sergio, César y Alonso, un ejemplo de demostración de que no es cuestión de puertas que se cierran, si no de las que puedes abrir e incluso crear. Jesus, una de las personas más fantásticas que he conocido y que me enseño como una persona puede hacerse valer y superar todo lo que le echen con orgullo e integridad. Borja y Salomé, tan lejos y a la vez tan cerca, enseñándome siempre con una cerveza en la mano esas noches en Deleitosa.

En tercer lugar quiero agradecer a mis profesores de la carrera por todo lo que me han enseñado. A Ricardo Imbert por guiarme para encontrar mi vocación. Al profesor Víctor Maojo por su atención y ofrecerme la gran oportunidad de mi carrera. En especial a mi tutor Máximo por la oportunidad de poder aprender todos los días bajo su tutela, de no rendirse conmigo, de aguantar todo lo pesado que puedo llegar a ser y por dejarme aportar un granito de arena al mundo con este trabajo. Álex, un compañero de trabajo increíblemente brillante; Guillermo y Ana, gran apoyo y ejemplo a seguir, veteranos donde los haya y los maestros de las soluciones. Y, finalmente, a todo el GIB por ser unos compañeros magníficos. No puedo olvidarme de todos los grandes amigos que he conocido en la facultad y me han alegrado tantas tardes de estudio: Jorge, Fran, Isma, Álvaro, Andi, Iván y Eu; los cuales no han dejado de impresionarme desde que los conozco.

Para finalizar la más importante, la persona que más quiero en este mundo, Cris; alguien que me completa y que no estaría aquí ni hubiese hecho todo lo que me ha llenado estos años de mi vida universitaria sin ella. Un apoyo incondicional la cual nunca voy a ser capaz de compensar todo lo que ha hecho y aún hace por mí.

Por todos vosotros y por más gente que no he podido nombrar por falta de espacio, sois unas personas por las que merece la pena guardar un recuerdo y derramar una lágrima de felicidad y esfuerzo.

Gracias.

Resumen

Diversos estudios confirman la rápida evolución en el uso y expansión de las tecnologías móviles en África. Esto supone una alternativa al uso de otros dispositivos menos compactos y menos disponibles. También se ha demostrado que las tecnologías móviles conectadas a la red proporcionan un apoyo educativo bastante efectivo. Por otro lado, la educación e investigación, en ciertos países del continente africano, es uno de los retos todavía pendientes a causa de una falta de recursos y medios.

Dada la situación podemos aprovechar esta tendencia para acercar la educación y la investigación a la comunidad africana. Además de su popularización, los dispositivos móviles han experimentado una mejora sustancial en sus capacidades en los últimos años. Hoy por hoy, las personas pueden realizar con un smartphone o tablet muchas de las tareas que podrían hacer en un ordenador de mesa. A esto hay que incluir la expansión de Internet, sus buenas implicaciones en el ámbito educativo y la globalización del uso del 3G en todo el mundo. Con todo esto, los dispositivos móviles son una posible herramienta alternativa para apoyar la educación y la investigación en África.

El presente Trabajo de Fin de Grado (TFG) se enmarca dentro del proyecto AFRICA BUILD. Dicho proyecto tiene como objetivo principal fomentar la investigación y fortalecer las capacidades de cuatro centros de educación superior en África a través de las TIC. Para cumplir este objetivo se ha diseñado el sistema AFRICA BUILD Portal (ABP). El ABP consiste en una solución web basada en “e-learning” para estudiantes, profesores e investigadores dentro del continente africano. El objetivo de este TFG consiste en ampliar la accesibilidad y el uso del portal a través de dispositivos móviles.

La solución propuesta en este TFG consiste en el desarrollo de dos soluciones que complementan el ABP: (I) una capa de servicios web para el portal y (II) la versión App para dispositivos móviles del mismo. Cabe destacar la relación existente entre ambas soluciones ya que la segunda necesita de la primera para poder funcionar y comunicarse así con el portal.

Las soluciones desarrolladas han cumplido los requisitos marcados y han supuesto un complemento importante para el proyecto. Los servicios web permiten la comunicación del portal con nuevas aplicaciones que deseen intercambiar información. La aplicación móvil ayuda a los usuarios del portal a continuar su labor alternando el uso entre sistemas sin comprometer su trabajo.

Abstract

Diverse studies confirm the increase of use and expansion of mobile technologies in Africa. This is an alternative for the use of other less compact and less available devices. Also, it has been demonstrated that mobile technologies connected to the network give an actual effective support. On the other hand, the education and researching, in certain African countries, it is one of the pendant challenges because a lack of resources.

At this point, we can use this tendency to approach education and researching to African community. Furthermore, mobile devices had experimented a great improvement in their functions in the last years. Nowadays, people can realize with a smartphone or tablet almost the same tasks as if they were working with a personal computer. We must include the Internet expansion, their great implications in education and the 3G globalization around the world. With all of this, mobile devices could be an alternative tool to give support to education and researching in Africa.

This Final Project Degree (FPD) is part of the AFRICA BUILD Project (AB). The main objective of AB is promote researching and fortify the capacities of four high school centres in Africa through Information and Communication Technologies. To reach this goal, it has been developed the AFRICA BUILD Portal (ABP). The ABP is an “e-learning” web solution for students, teachers and researchers of Africa. The main goal of this FPD is to extend the accessibility and use of the portal through mobile devices.

The proposed solution consist on the development of two solutions that complements the ABP: (I) a web service layer for the portal and (II) the App version for mobile devices. We must highlight the relationship between both solutions because the second one needs the first to be able to work and communicate with the portal.

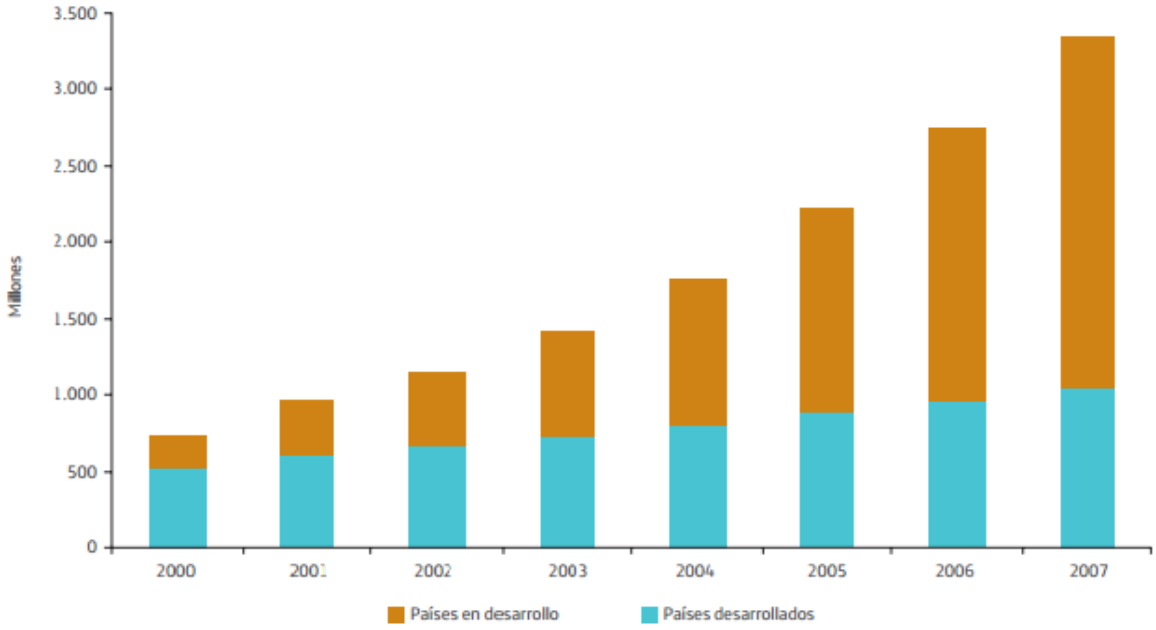
The developed solutions had reached the requirements set. They are an important complement for the project. The web services allow the communication of the portal with another new applications. The App helps users to continue his work rotating between both systems without compromise his work.

Tabla de contenido

1. INTRODUCCIÓN Y OBJETIVOS

1.1. Motivación

En el ámbito de la educación y el desarrollo, diversas investigaciones y publicaciones muestran las buenas implicaciones del uso de Internet y, más concretamente, las tecnologías móviles [1] [2]. Con respecto a estas últimas, se ha contemplado un incremento masivo del uso de los dispositivos móviles ya desde hace más de diez años [3] [4]. Los estudios referenciados analizan y justifican las causas que lo han provocado. Una de las razones a destacar es la mayor facilidad que poseen los dispositivos móviles en conectividad y acceso a la red [5], especialmente en lugares desfavorables o con pocos recursos, los cuales tienen difícil el establecimiento, mantenimiento y acceso de equipos informáticos y/o de comunicación [6].



Fuente: Unión Internacional de Telecomunicaciones (UIT). World Telecommunication/ICT Indicators Database.

Fig. 1 Número de suscripciones a móviles en los países desarrollados y en los países en vías de desarrollo (2000 a 2007) [1].

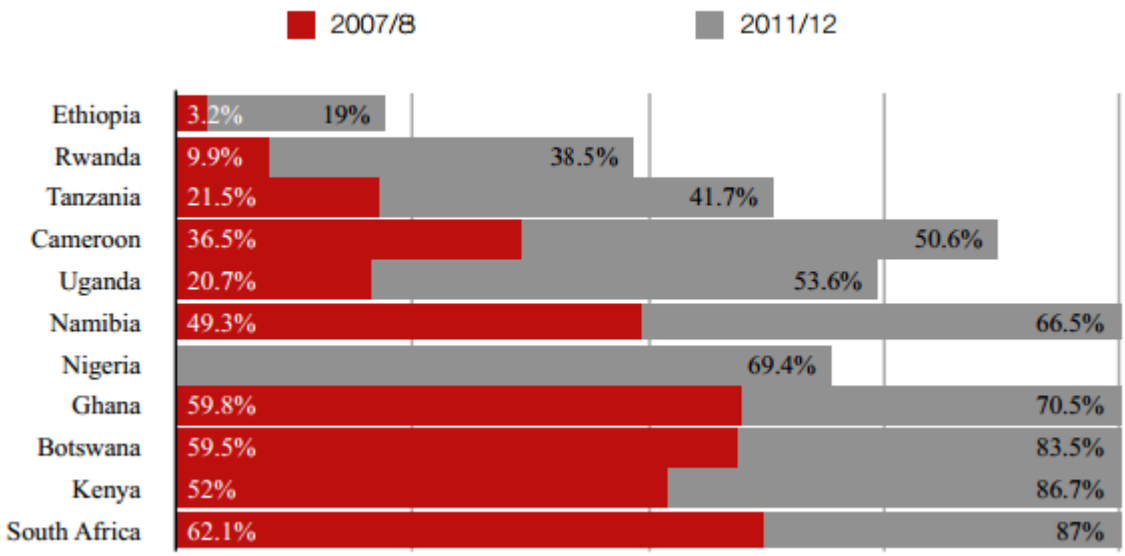


Fig. 2 Muestra de usuarios mayores de 15 años con dispositivo móvil o tarjeta SIM activada [5].

Con respecto al incremento del uso de dispositivos móviles y acceso a Internet, los estudios muestran un mayor crecimiento en países en desarrollo (Fig. 1). Dicho incremento se lleva contemplando desde hace una década en adelante y confirman la profunda inclinación de la población africana en el uso de la tecnología móvil (Fig. 2). Además, la tecnología 3G está completamente expandida desde hace unos años, favoreciendo así el acceso a la red (Fig. 3).

Una vez analizada la situación de la evolución en el acceso a Internet, se plantea aprovechar dicho avance para ayudar de alguna forma en la educación e investigación de la comunidad africana, las cuales están íntimamente ligadas. Sin la investigación no puede ampliarse el conocimiento y, a su vez, la educación es necesaria para poder realizar actividades de investigación.

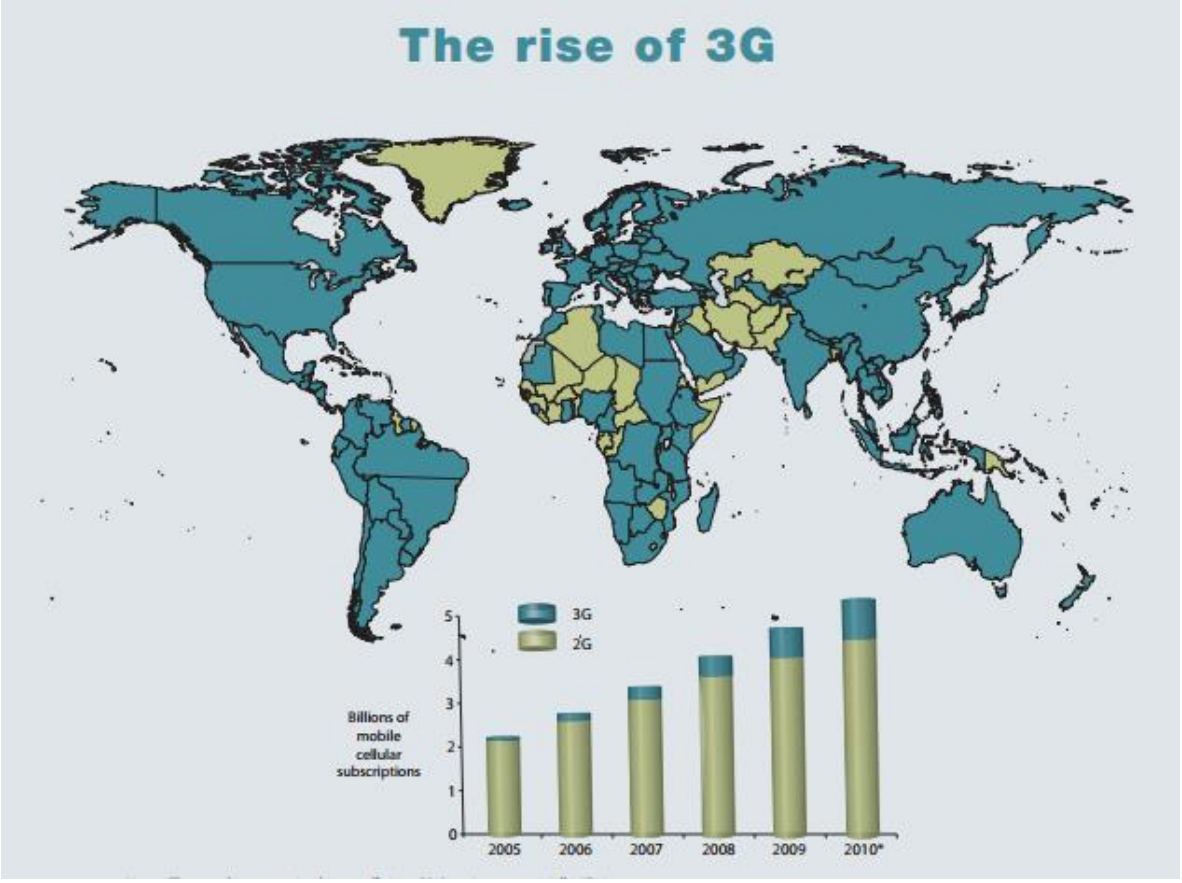


Fig. 3 Subscripciones móviles en el 2010 [7]

En una de las publicaciones de 2009 de la revista "Nature" se afirma la aparición de una clara diferencia en cuanto al número de publicaciones entre los países desarrollados y los que están en vías de desarrollo (Fig. 4). La justificación de este hecho consiste en la falta de recursos o prioridades educativas en muchos de estos países a causa de diversos problemas económicos y sociales, los cuales, junto a la falta de recursos, merman aún más la capacidad de dedicación a la investigación, educación y salud [8].

Podemos concluir que el uso de las tecnologías de la información, la educación y la investigación poseen el mismo problema subyacente: La falta de acceso a los recursos. Cada uno de los tres conceptos mencionados puede potenciar a los demás. En el presente TFG, se propone una solución que fomente la educación e investigación a través de una mejora en el acceso a recursos educativos mediante el uso de dispositivos móviles conectados a la red.

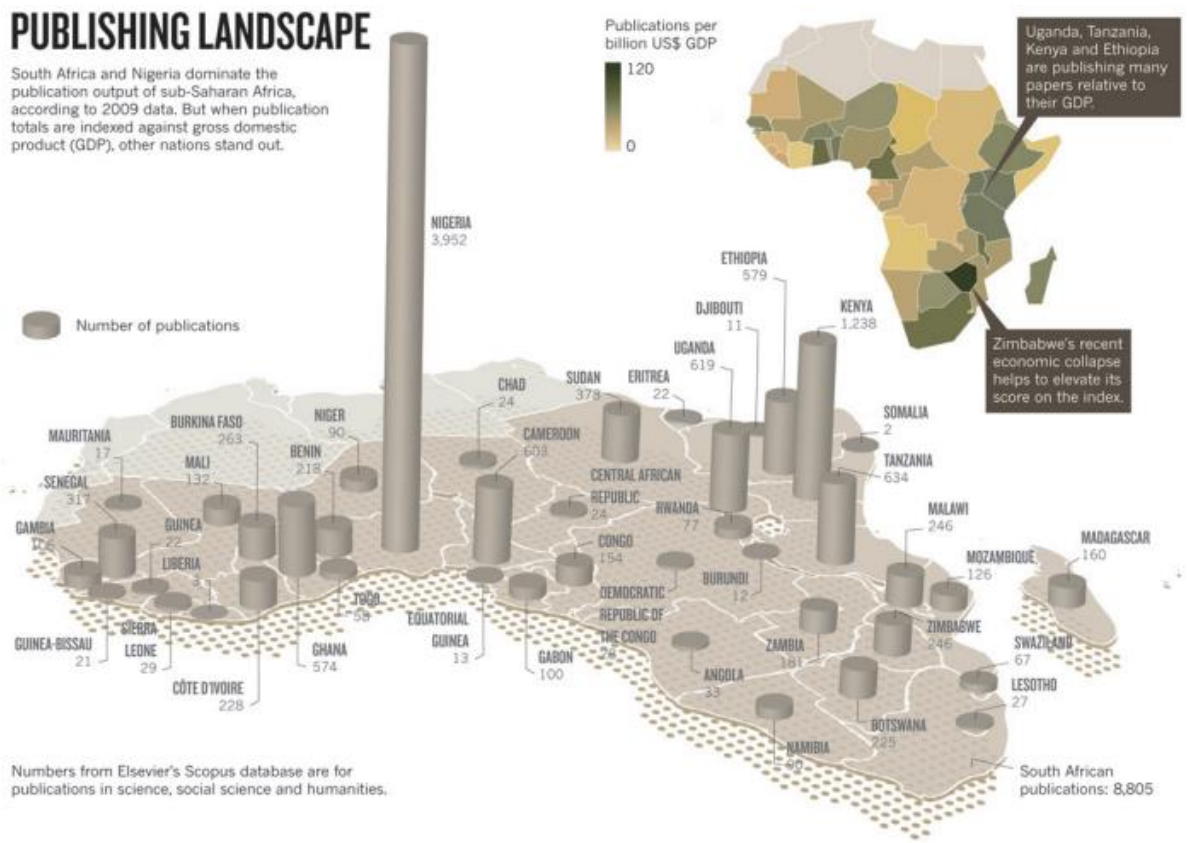


Fig. 4 Número de publicaciones realizadas en África en 2009 [8]

Por otro lado, durante los últimos años han surgido nuevos conceptos asociados a la educación online: El "e-learning" y, más concretamente, el "m-learning" (a través de dispositivos móviles) [9]. Dichos conceptos aparecen no solo por el aumento de la capacidad de acceso a la red. Para el caso del "m-learning", se debe incluir también la evolución y mejora de las prestaciones y las capacidades de los dispositivos móviles [10]. Recientemente, dichos dispositivos proporcionan unas capacidades de interacción, accesibilidad, usabilidad y visualización de los recursos más que suficientes y adecuados para el tratamiento de contenido educativo [11].

El "m-learning" se define como una metodología de enseñanza y aprendizaje valiéndose del uso de dispositivos móviles [9] como smartphones, móviles y tabletas. Las tecnologías móviles se han extendido en todos los dominios y prácticamente se puede hacer cualquier cosa con un dispositivo móvil o tablet en vez de un equipo informático de mesa o portátil [10]. Entre las capacidades de un dispositivo móvil podemos destacar la libertad y flexibilidad para el aprendizaje, la mejora en el acceso a datos y recursos de manera casi inmediata [12], proporcionar herramientas de trabajo en equipo y material de apoyo como textos, videos, test, etc… [13].

Para la introducción a la población del uso del "m-learning", diversas instituciones han realizado proyectos educativos con el objetivo de fomentar la educación a través de dispositivos móviles [12]; entre ellas, "From E-learning to M-learning" [14] de Ericsson, "La guía del Mobile Learning" de la fundación Telefónica [15] y la Primera Semana del Aprendizaje Móvil (MLW) [16] por la UNESCO.

A modo de referencia, existen diversas aplicaciones móviles educativas. Duolingo [17], por ejemplo, consiste en una App de aprendizaje de idiomas. La aplicación ayuda en la gramática, lectura y pronunciación del idioma a través de quizzes, unir palabras, micrófono, completar frases, etc… y ejercicios de refuerzo para no olvidar los conceptos, todo ello apremiado con logros, puntos, niveles y experiencia del usuario, entre otros. Otro ejemplo puede ser la App Clan de RTVE [18]. En ella, los niños pueden ver contenido didáctico con el móvil y con la posibilidad de visualizarlo en inglés. Existen además, otras aplicaciones "m-learning" con características sociales [19] como foros de discusión, redes sociales, comentarios en Twitter, etc…

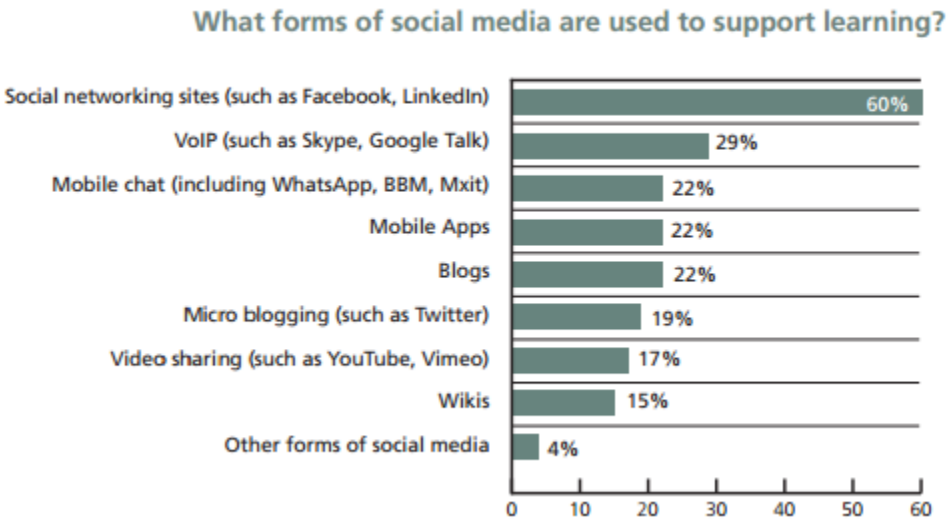


Fig. 5 Estudio sobre el uso de diferentes formas de medios de aprendizaje en África [20].

En el caso concreto de la comunidad africana, las actividades asociadas al "e-learning" y al "m-learning" están siendo muy bien acogidas, según algunas de las encuestas y estudios realizados [20] [21], los cuales indican una clara actitud positiva en el uso y utilidad de dicho método de aprendizaje.

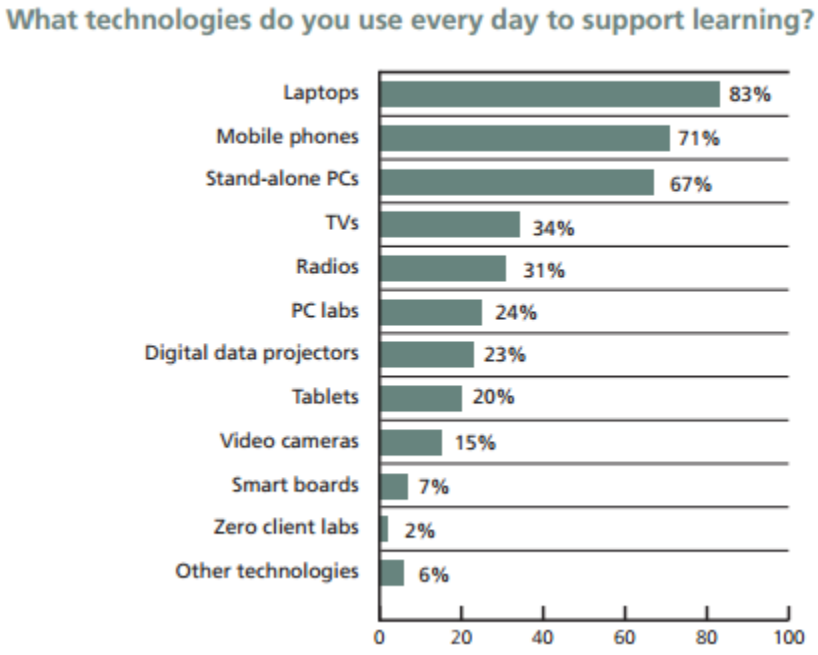


Fig. 6 Estudio sobre el uso de tecnologías y dispositivos aplicados al aprendizaje en África [20].

Podemos afirmar entonces las buenas implicaciones del "m-learning" y, para el caso de África, el uso de Internet y los dispositivos móviles pueden ser una buena ayuda en el desarrollo de la educación y la investigación.

En este contexto se ubica el proyecto AFRICA BUILD, donde se enmarca este Trabajo de Fin de Grado (TFG). El presente TFG propone una solución basada en "m-learning". Dicho sistema de comunicación, basado en servicios web, es necesario para la correcta cohesión de la solución propuesta y el proyecto.

En las siguientes secciones se describe el proyecto AFRICA BUILD donde se ubica el TFG, en qué consisten las soluciones a desarrollar, su contexto, objetivos y de qué manera se relacionan con el proyecto. Seguidamente, se desglosa todo el ciclo de vida de las soluciones propuestas: Requisitos, Diseño, Implementación y Plan de pruebas. Finalmente, analizaremos los resultados, las conclusiones y las posibles líneas futuras que surgen tras la finalización del proyecto.

1.2. Contexto

1.2.1. Proyecto AFRICA BUILD

AFRICA BUILD es una Acción de Coordinación que pretende fomentar la investigación y educación sanitaria en África, a través de las Tecnologías de la Información. Este proyecto está financiado por el Séptimo Programa Marco de la Unión Europea (FP7-ICT). AFRICA BUILD comenzó el 1 de Agosto

de 2011 y durará un periodo de 36 meses [22]. El proyecto está coordinado por el Grupo de Informática Biomédica de la ETSIInf.

El proyecto está siendo desarrollado en colaboración con las siguientes instituciones mundiales, europeas y africanas:

- Instituto de Medicina Tropical de Bélgica.
- Instituto de Información Tecnológica de Egipto.
- Facultad de Medicina-Farmacia y odontoestomatología, Universidad de Bamako, Mali.
- Universidad de Ginebra, Suiza.
- Organización Mundial de la Salud.
- Facultad de Medicina y de las ciencias Biomédicas de Camerún.
- Escuela de Salud pública, Universidad de Ghana.

Mientras que en un principio se pretendía que los socios europeos dieran soporte y transfirieran el conocimiento necesario a los socios africanos para satisfacer sus necesidades, el fin último del proyecto es la generación de comunidades virtuales de investigadores africanos que puedan continuar con los esfuerzos iniciales creando, desarrollando e intercambiando, colaborativamente, nuevo conocimiento, métodos, herramientas informáticas y datos. AFRICA BUILD pretende alcanzar este objetivo a través del uso de las nuevas tecnologías, favoreciendo la colaboración entre investigadores que puedan compartir recursos de investigación sanitaria.

Desde una perspectiva conceptual, AFRICA BUILD proveerá el soporte científico, tecnológico y financiero para el desarrollo de la excelencia en centros de educación e investigación en salud en África. Estas actividades fomentarán las capacidades y la excelencia científica en centros africanos y será el punto de partida para futuros desarrollos colaborativos que asegurarán sostenibilidad una vez el proyecto concluya.

1.2.2. AFRICA BUILD Portal

Uno de los objetivos fundamentales de este proyecto es el desarrollo de una plataforma, conocida como AFRICA BUILD Portal, orientada a fomentar la investigación y educación sanitaria entre estudiantes, investigadores y profesores africanos. El portal permite acceder a diferentes tipos de recursos relacionados con la medicina y la salud —i.e. blogs, cursos, documentación, foros, wikis, etc… El fin último de esta plataforma es fomentar la colaboración entre los investigadores africanos, con el fin de crear redes sur-sur que impulsen la investigación hecha por y para africanos.

Con respecto al AFRICA BUILD Portal (ABP), la plataforma es accesible vía web y proporciona un conjunto de herramientas tanto educativas y de investigación como sociales, convirtiéndolo así en una red social enfocada a la investigación sanitaria y la colaboración entre personas, un marco que ayuda a incrementar la investigación en países de África.

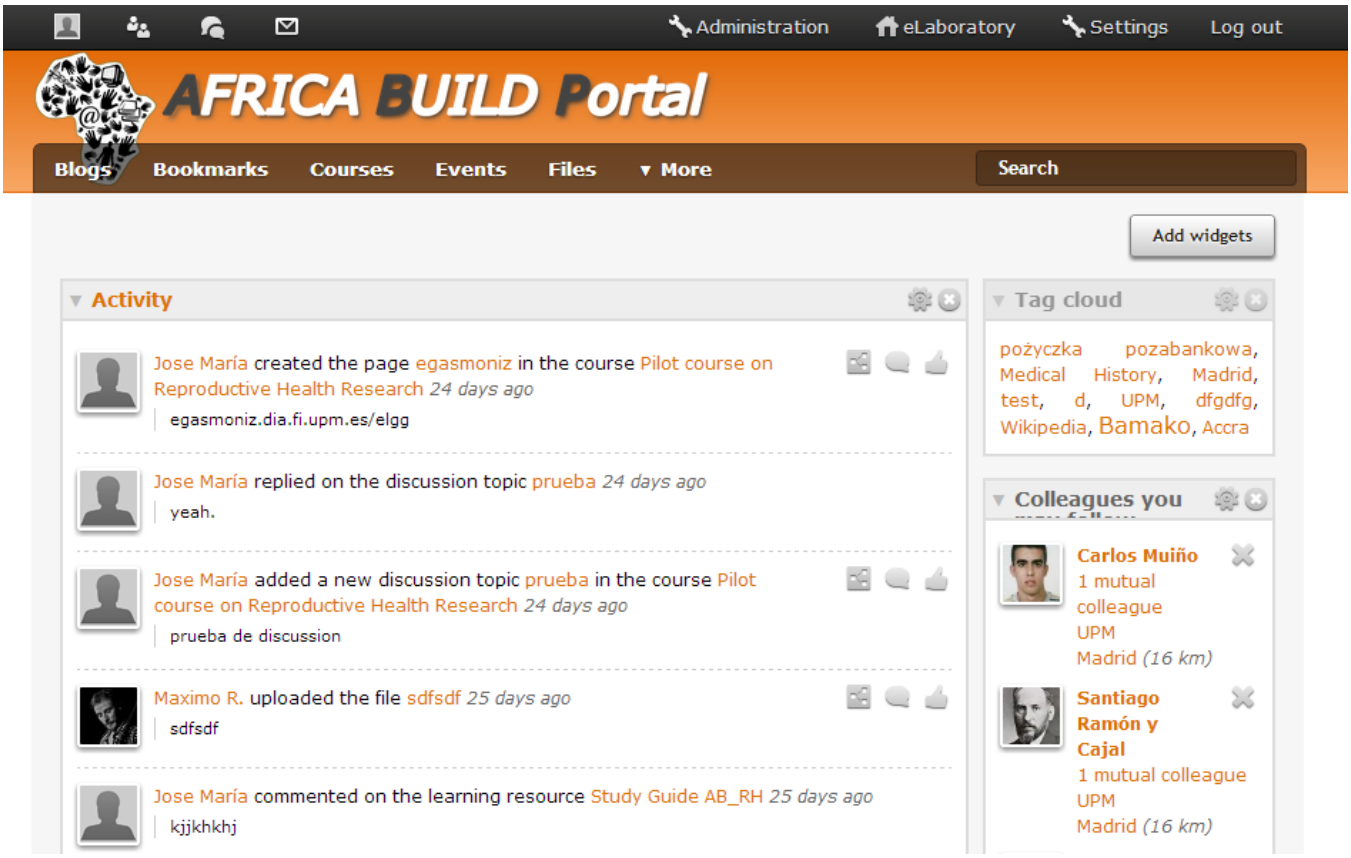


Fig. 7 Página principal del AFRICA BUILD Portal en desarrollo

El portal se fundamenta en tres pilares:

- E-learning: la red supone uno de los elementos más útiles a la hora de apoyar y fomentar la educación, el desarrollo y la investigación en todos los ámbitos o "wikis", blogs, publicaciones científicas, páginas web de interés científico o educativo, etc…
- Investigación: es fundamental a la hora de generar nuevo conocimiento y es otra forma más de estudio y formación. La dedicación a este pilar puede romper barreras y abrir nuevas puertas hasta ahora desconocidas para estudiantes y demás personas que buscan el progreso allá donde estén.
- Colaboración: el uso de la red como marco común de trabajo para los usuarios ayuda en muchas situaciones, en especial la disposición y disponibilidad de los medios y recursos, así como establecer grupos de trabajo correctamente comunicados y eficaces con un objetivo en común.

1.3. Planteamiento del problema

Actualmente el AFRICA BUILD Portal (ABP) no posee una adaptación para su uso a través de dispositivos móviles. El ABP es una solución basada en web 2.0, sin embargo, aunque dichas

soluciones sean accesibles a través de un smartphone o tablet, el uso del portal contiene problemas de visualización y usabilidad para pantallas táctiles de tamaño reducido. Esto puede suponer un gran impedimento para su acceso y utilización. Hay que añadir que el problema se intensifica en países donde predomina el uso de dispositivos móviles en detrimento de dispositivos fijos como ordenadores de mesa o portátiles [5] [23].

Por otro lado, surge un problema a la hora de realizar una adaptación ajena al portal. El nuevo sistema debe ser acorde al uso del portal simulando la visualización, estilos, contenido, etc… Dicho esto, la adaptación utilizará la funcionalidad que le ofrezca el ABP, respetando en todo momento las restricciones y los objetivos del mismo. Además, el futuro usuario no debe ser capaz de distinguir entre la solución web ya desarrollada y la nueva solución. Para ello, se debe realizar una comunicación entre sistemas que permitan la coherencia y sincronización de tal manera que el usuario pueda alternar entre los sistemas a su gusto sin ello comprometer su trabajo.

A modo de referencia, plataformas como Twitter o Facebook ofrecen una capa de servicios con funcionalidad expuesta. Dichos servicios sirven para que otros sistemas puedan sincronizarse con estas plataformas, obtener información y datos de las mismas. De este modo los desarrolladores diseñan sus propias aplicaciones en comunicación con las plataformas a través de los servicios. De este modo, tanto la plataforma como el tercero se comunican y colaboran mutuamente en su difusión, utilidad y uso. Para comprender mejor la cohesión de estos conceptos, se definen los dos pilares que fundamentan la solución propuesta a los objetivos de este TFG: *M-learning* y los *Servicios Web*. Dichos puntos son descritos en el apartado 2.

1.4. Objetivos

El diseño y desarrollo de la versión móvil del ABP parte como objetivo principal, proporcionando así una alternativa de acceso a la plataforma y sus recursos, de tal manera que un usuario no necesite necesariamente un equipo de mesa para poder realizar sus tareas relacionadas con el portal. También forma parte de los objetivos principales la integración del módulo que se encarga de exponer los recursos, servicios y la funcionalidad del portal al exterior.

Como objetivos secundarios asociados al trabajo se enumeran los siguientes:

- Diseñar una correcta adaptación de la visualización de los recursos para dispositivos de pantalla reducida.
- Ampliar el abanico de disponibilidad de la instalación posible para todos los dispositivos móviles, smartphones, tablets, entre otros.
- Devolver al usuario un “feedback” a modo de respuesta por el uso de los recursos y el envío de posibles respuestas o interacciones con los mismos a través de la red.
- Enfatizar en el concepto social de la plataforma para fomentar la colaboración entre los participantes.
- Proporcionar una base común para futuras implementaciones y sistemas que complementen o necesiten acceso a los recursos del portal.
- Diseñar la funcionalidad descrita respetando la correcta sincronización y consistencia entre los datos del portal y los almacenados fuera del mismo.

1.5. Solución Propuesta

Se propone un paquete de capacidades que puedan proporcionar la funcionalidad el ABP a través de dispositivos móviles. Para ello se decide diseñar una aplicación móvil bautizada como **AFRICA BUILD APP** con las siguientes características:

- Disponibilidad para la plataforma **Android**: uno de los sistemas operativos móviles más extendidos en el planeta en una gran variedad de dispositivos. Posee una API muy completa que proporciona, entre otros, comunicación con otras aplicaciones, y su integración con dispositivos de interacción y/o comunicación como pantalla táctil, cámara, GPS, etc....
- Acceso a los **recursos de aprendizaje** que ofrece el portal como: cursos, documentos, videos, test, blogs, foros, wikis, etc…
- Funcionalidades y herramientas sociales para la visualización de la actividad y, en resumen, la integración del usuario dentro de la comunidad del portal.

Se decide desarrollar, en conjunción con la aplicación, una capa de servicios Web del portal para la comunicación entre aplicaciones externas (entre ellas la App) y el portal. Dicha capa posee las siguientes propiedades:

- Arquitectura basada en el modelo **REST/RPC híbrido**: interfaz web simple que utiliza HTTP para el acceso a funciones remotas sin las abstracciones adicionales de los protocolos basados en patrones de intercambio de mensajes como el protocolo de servicios web SOAP.
- Documentación que define y describe toda la funcionalidad de la capa, así como los recursos definidos, y esquema de las respuestas proporcionadas.
- Canal común de comunicación basado en el intercambio de datos basado en XML o JSON entre el portal y los sistemas externos.

La idea global que unifica las dos propuestas consiste en lo siguiente: tanto la aplicación móvil como las demás aplicaciones y/o sistemas que se implementen en un futuro se comunican con la capa de servicios Web para la interacción con el portal. Todo sistema externo que desee establecer una comunicación con el ABP deberá conocer los esquemas de envío y recepción de datos que define la capa de servicios Web (documentación) para poder tratar usar la información contenida (ver Fig.7).

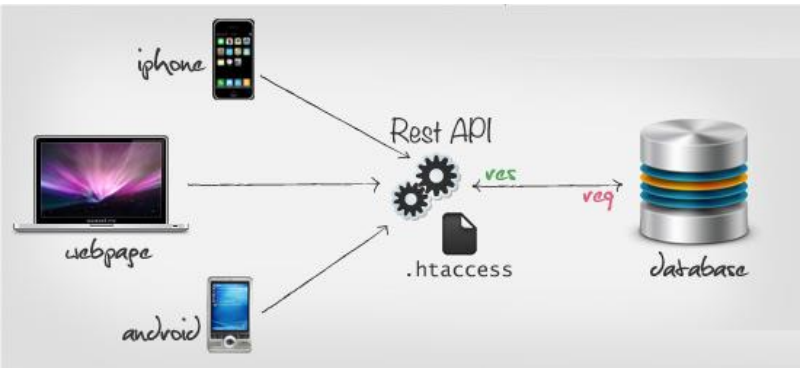


Fig. 8 Diferentes dispositivos comunicados con una base de datos a través de una API REST.

2. ESTADO DEL ARTE

En este apartado se analiza el estado de la cuestión, con el objetivo de encontrar la solución más óptima y efectiva. A continuación se desglosan todos los conceptos que han sido necesarios durante el desarrollo de este trabajo..

2.1. Redes Sociales

Una red social es la representación de una estructura (social) mediante un grafo. Dicho grafo se compone de la relaciones entre individuos dentro de la red. Cada relación construye una línea que conecta los nodos que representan a los individuos. El tipo de conexión representable en una red social consiste en un lazo interpersonal, que se puede interpretar como relaciones de amistad, parentesco, laborales, entre otras [24].

Varios ejemplos representativos de redes sociales (servicios) son Facebook, Google+ y Linkedin. Además, la filosofía de las redes sociales está empezando a ser utilizada en sistemas previos como YouTube, con su nuevo sistema de actividades, o Twitter y su módulo de "Microblogging".

2.1.1. Redes sociales en la educación

Diversos estudios sobre educación determinan que las redes sociales pueden convertirse en una herramienta de aprendizaje e intercambio de información que fomente también la cooperación [25] [26]. Los fines de una red social, de cara a los participantes, pueden incluir un refuerzo académico a través de la red, integrando actividades que tengan que ver con el uso de email, foros, chats, mensajería, blogs, cuestionarios, evaluaciones interactivas, etc. [25]. Por otro lado, actividades en las que los estudiantes formen parte de investigaciones o juegos pueden facilitar el acceso a temas y/o recursos de su interés [26]. Aumenta así el sentimiento de comunidad educativa y mejora el ambiente de trabajo al permitir a los alumnos crear sus propios objetos de interés [26].

Podemos concluir, junto con la experiencia personal, que las redes sociales pueden enfocarse al ámbito educativo de una manera fluida, práctica y entretenida, contribuyendo así al aprendizaje y a la comunicación entre usuarios de todo tipo (docentes, estudiantes, investigadores, etc…). Esta contribución es reforzada cuanto mayor sea el número de participantes [25], ya que a mayor sea la comunidad, más enriquecedor es el intercambio de conocimiento la participación.

2.2. Aplicaciones Móviles

Conocido comúnmente como *App*, consiste en una aplicación informática diseñada para ser ejecutada en dispositivos móviles. Por lo general, las Apps se encuentran disponibles a través de plataformas de distribución. Estas plataformas son gestionadas por las compañías propietarias de los sistemas operativos móviles (App Store y Google Play, entre otros).

El desarrollo de aplicaciones para dispositivos móviles requiere tener en cuenta las limitaciones de estos dispositivos. Un ejemplo es que estos dispositivos funcionan con batería y tienen procesadores menos potentes que los ordenadores personales [27]. Las implementaciones de las Apps también tienen que considerar una gran variedad de tamaños de pantalla, datos específicos de software y configuraciones. Por ejemplo, para una plataforma de diseño de una App se impone el

lenguaje de desarrollo a usar: Java para Android [28], Swift para IOS [29] o C# para Windows Phone [30].

2.2.1. M-learning

También conocido como Aprendizaje Electrónico Móvil, se define como una metodología de enseñanza y aprendizaje valiéndose del uso de dispositivos móviles. Estos dispositivos pueden ser teléfonos móviles, PDA, tabletas, smartphones y todo dispositivo de mano que tenga alguna forma de conectividad inalámbrica [13]. El "mobile learning" o "m-learning" tiene ventajas pedagógicas sobre otros modelos educativos, incluso sobre su predecesor "e-learning". Entre las ventajas principales radica la capacidad de ofrecer un aprendizaje personalizado en cualquier momento y lugar, la posibilidad de llevar a cabo un aprendizaje adaptado a cada estudiante y el dinamismo con el que se presenta para los alumnos un medio tan atractivo como este [9].

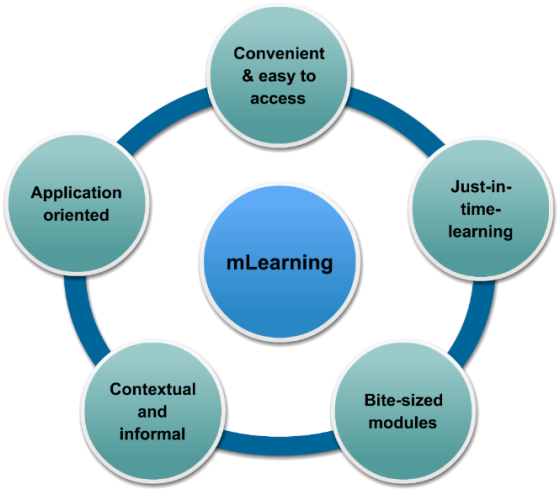


Fig. 9 Características del M-learning [31]

2.3. Servicios Web

Se define como un componente software que utiliza un conjunto de protocolos y estándares para intercambiar datos entre aplicaciones sobre una red [32]. La idea de este concepto consiste en exponer la funcionalidad que un sistema o servidor en la red desea publicar y que pueda ser tratada desde el exterior, con lo que permite expandir y ampliar tanto la utilidad como la accesibilidad al mismo [32]. Los servicios web ayudan a la comunicación entre sistemas con grandes diferencias arquitectónicas, de tal manera que se abstraen de su lógica y estructura interna y pueden tener un marco común para el intercambio de comunicación.

Fig. 10 Comunicación entre diferentes sistemas vía Servicios Web.

Los servicios Web ofrecen una gran flexibilidad en diseño e implementación. No dependen de ningún sistema operativo en particular o de una aplicación anfitriona, dando la libertad de elegir o usar cualquiera que se adapte mejor a las necesidades del sistema. A continuación se describen las dos arquitecturas de servicios web más extendidas en la actualidad: SOAP, REST y RPC.

2.3.1. Sistemas Orientados a Servicios

Conocida como Arquitectura Orientada a Servicios o SOA (del inglés *Service Oriented Architecture*), consiste en un paradigma de arquitectura para diseñar y desarrollar varios tipos de Sistemas Distribuidos [33]. Las soluciones SOA han sido creadas para satisfacer los objetivos de negocio. Estas incluyen facilidad y flexibilidad de integración con otros sistemas heredados, reduciendo los costes de implementación y una adaptación ágil ante cambios [34].

Al contrario de las arquitecturas orientadas a objetos, las SOA están formadas por servicios de aplicación débilmente acoplados y altamente interoperables. Para comunicarse entre sí se utiliza un lenguaje independiente de la plataforma que lo soporta, fomentando así la abstracción del sistema [34]. Consiste pues en la definición de un módulo de interfaz que oculta las particularidades de una implementación, lo que la hace independiente del fabricante, del lenguaje de programación o de la tecnología de desarrollo. Con esta arquitectura, se pretende que los componentes de software desarrollados sean muy reutilizables y cohesionables.

2.3.1.1. SOAP

SOAP se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y que tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP, etc… y SOAP se encarga del formato de los mensajes [35]. El mensaje SOAP está compuesto por un sobre (del inglés, *envelope*), cuya estructura está formada una cabecera y cuerpo (del inglés, *header* and *body*).

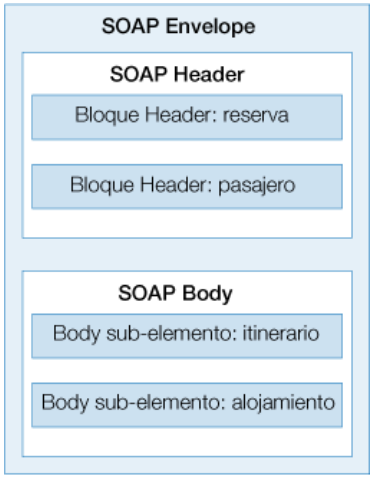


Fig. 11 Estructura de un mensaje en SOAP

Para optimizar el rendimiento de las aplicaciones basadas en Servicios Web, se han desarrollado tecnologías complementarias a SOAP, que agilizan el envío de los mensajes (MTOM) y los recursos que se transmiten en esos mensajes (SOAP-RRSHB) [35].

2.3.1.2. REST

Los Servicios Web basados en REST intentan emular al protocolo HTTP o protocolos similares mediante la restricción de establecer la interfaz a un conjunto conocido de operaciones estándar (GET, PUT, POST y DELETE) [36]. Este estilo se centra más en interactuar con recursos con estado, que con mensajes y operaciones. El término fue introducido en la tesis doctoral de Roy Fielding en 2000, quien es uno de los principales autores de la especificación de HTTP [36].

En realidad, REST se refiere a una colección de principios para el diseño de servicios web. Estos principios resumen cómo los recursos son definidos y diseccionados. Es importante destacar que REST no es un estándar, sino un estilo de arquitectura basado en estándares como HTTP, URL, estándares de representación de recursos (XML, HTML, GIF y JPEG entre otros), etc…

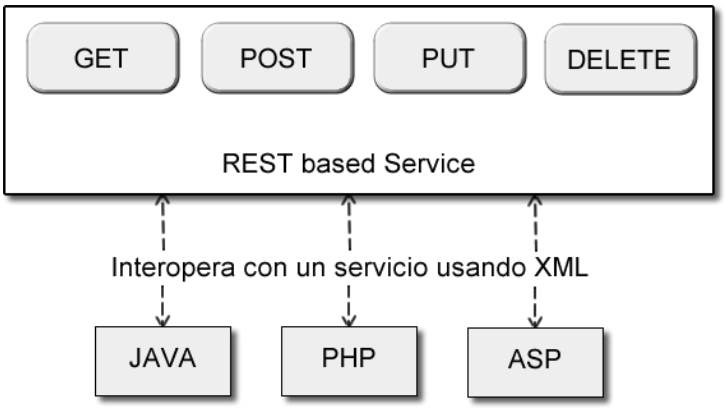


Fig. 12 Esquema REST

2.3.2. RPC

Definido como *Remote Procedure Call* del inglés o Llamada a Procedimiento Remoto. Definido en 1984 por Andrew D. Birrell y Bruce Jay Nelson, RCP consiste en un protocolo que permite la ejecución de código localizado en sistemas remotos [37]. El protocolo permite abstraer al sistema remoto de la comunicación utilizada con el sistema objetivo. RCP es bastante común para la comunicación cliente-servidor. El cliente realiza una llamada a una función en un sistema remoto; el servidor recoge la petición, realiza la lógica de la función y envía el resultado al cliente [38].

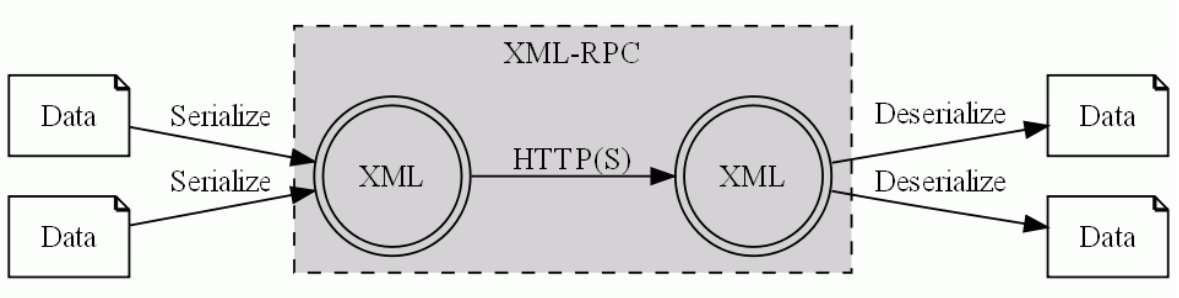


Fig. 13 Ejemplo de comunicación basada en XML-RPC

El intercambio de información debe hacerse a través de un lenguaje común, el cual entienden tanto el emisor como el receptor. El objetivo de esto es abstraer la información del lenguaje y la arquitectura de los sistemas en comunicación.

El protocolo RCP fue una de las primeras arquitecturas de servicios web. Puede decirse que las primeras herramientas centradas en esta visión formaban parte de la primera generación de servicios web. El protocolo no forma parte de los SOA ya que define la operación como la unidad básica de comunicación, en vez del mensaje. Mediante un proceso denominado serialización (*marshalling* en inglés) [39], los datos son adaptados a un formato antes de viajar a través del canal. El emisor se encarga de formatear o serializar la información a enviar mientras que el receptor deserializa la respuesta para su uso (Fig. 13). Los formatos más utilizados son XML y JSON y el protocolo de comunicación comúnmente usado es HTTP.

3. TECNOLOGÍAS EMPLEADAS

En este apartado se describen todas las tecnologías y lenguajes implicados en el desarrollo de la solución.

3.1. Lenguajes

3.1.1. Java

El lenguaje de programación Java fue originalmente desarrollado por James Gosling de Sun Microsystems (adquirida por la compañía Oracle) y publicado en 1995 [40]. Es un lenguaje de programación de propósito general, concurrente, orientado a objetos y basado en clases que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible.

Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (WORA, o "write once, run anywhere") [41], lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. El lenguaje Java se creó con el objetivo de facilitar el desarrollo de programas orientados a objetos y poder ser ejecutados en cualquier sistema operativo [41].

Para ello, el código fuente compilado genera un código conocido intermedio conocido como “bytecode”. Dicho bytecode es ejecutado en una máquina virtual (JVM), un programa escrito en el código de la plataforma anfitriona (que es el que entiende el hardware del sistema), que interpreta y ejecuta el código a compilar [41].

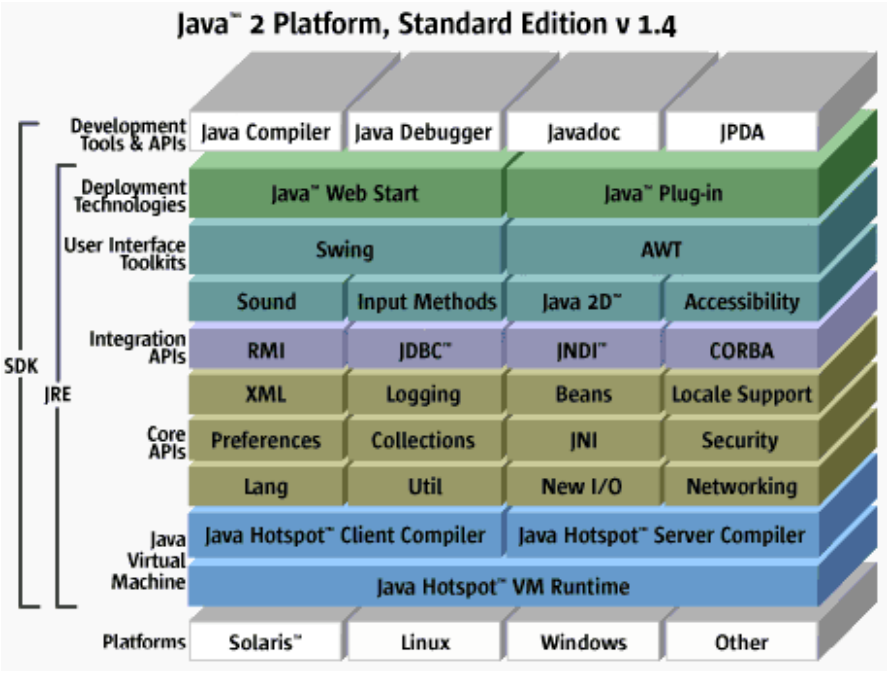


Fig. 14 Arquitectura de la plataforma Java [41]

3.1.2. HTML

HTML, del inglés *Hyper Text Markup Language* (Lenguaje de marcación de Hipertexto) es el lenguaje de marcas de texto utilizado por antonomasia en la Web y un estándar más a cargo de la W3C [42], organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo a su escritura e interpretación. HTML fue creado en 1986 por el físico nuclear Tim Berners-Lee [42]. Berners-Lee tomo dos herramientas preexistentes: El concepto de Hipertexto o *link,* el cual permite conectar dos elementos entre sí, y el SGML (Lenguaje Estándar de Marcación General) que sirve para colocar etiquetas o marcas en un texto que indique su visualización [42]. HTML no es propiamente un lenguaje de programación como C++, Visual Basic o Java, sino un sistema de etiquetas. El lenguaje no presenta ningún compilador, por lo tanto algún error de sintaxis que se presente éste no lo detectará y se visualizara en la forma como éste lo entienda.

```
    <div id="menucontainer">
        <h1>Info about <b>Product 1</b></h1>
    </div>
    <div id="leftArea">
        <h4>Select product</h4>
        <div class="productList">
            <div><a href="/linkHere">Product 1</a></div>
            <div><a href="/linkHere">Product 2</a></div>
            <div><a href="/linkHere">Product 3</a></div>
        </div>
    </div>
</div>
<div id="main">
    <table>
        <tbody>
            <tr><td>Name: </td><td>Product 1</td></tr>
            <tr><td>Description:</td><td id="description">Lorem ipsum dolo
            <tr><td>Price: </td><td id="price">CHF <b>100</b></td></tr>
            <tr><td></td><td><input type="button" value="BUY NOW!!!" /></t
```

Fig. 15 Ejemplo de código HTML

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo, script, pdf, etc...) se hace una referencia a la ubicación de dicho elemento mediante texto. La página web contiene sólo texto mientras que el navegador web interpreta el código, une todos los elementos y muestra la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma por cualquier navegador web actualizado.

3.1.3. PHP

Creado en 1994 por Rasmus Lerdorf, PHP del inglés *Hypertext Preprocessor*, es un lenguaje de "scripting" de propósito general pensado para el desarrollo web y embebido en páginas HTML [43].

Su sintaxis es similar a lenguajes como C o Java. La meta principal de este lenguaje en un principio es permitir a los desarrolladores web escribir de manera dinámica y rápida páginas web.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>

        <?php
            echo "¡Hola, soy un script de PHP!";
        ?>

    </body>
</html>
```

Fig. 16 Ejemplo de código PHP en un documento HTML [43]

En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), los ficheros .php contienen HTML con código PHP incrustado (en este caso, mostrar "¡Hola, soy un script de PHP!). El código de PHP está encerrado entre las etiquetas especiales de comienzo y final <?php y ?> que permiten entrar y salir del "modo PHP" [43]. El código en PHP es ejecutado en el servidor, generando HTML y enviándolo al cliente. El cliente recibirá el resultado de ejecutar el script sin saber el código que lo ha generado. El servidor web puede ser incluso configurado para que procese todos los ficheros HTML con PHP, por lo que no hay manera de que los usuarios puedan saber qué código existe detrás [43].

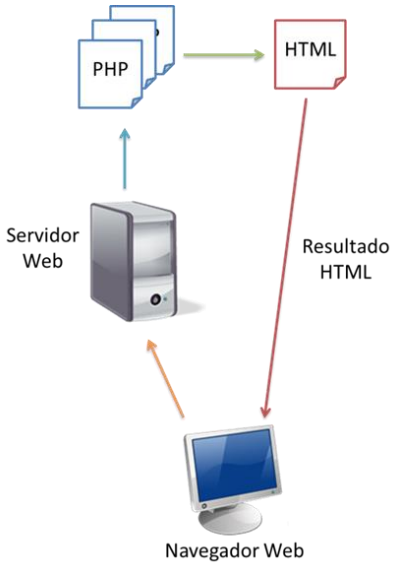


Fig. 17 Funcionamiento de PHP en un servidor

3.1.4. SQL

SQL, del inglés *Structured Query Language*, es un lenguaje para generar, manipular y recuperar información de una bases de datos relacionales. El modelo relacional de una base de datos se basa en la lógica de predicados y teoría de conjuntos, donde toda la información es clasificada en entidades, atributos de entidades y relaciones entre entidades. Una de sus capacidades es el uso de consultas o "querys" para recuperar, editar, crear o borrar información de bases de datos relacionales [44].

SQL tiene, entre otros, las siguientes características [45]:

- Lenguaje de definición de datos: El LDD de SQL proporciona comandos para la definición de entidades y relaciones, generando así el esquema completo de la base de datos.
- Consultas de manipulación de datos: El LMD de SQL incluye lenguajes de consultas basado en lógica de predicados para el tratamiento de los datos.
- Integridad: El LDD de SQL incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados.
- SQL incorporado ("embedded"): Se pueden incorporar instrucciones de SQL en otros lenguajes de programación como: C++, C, Java, PHP, etc…

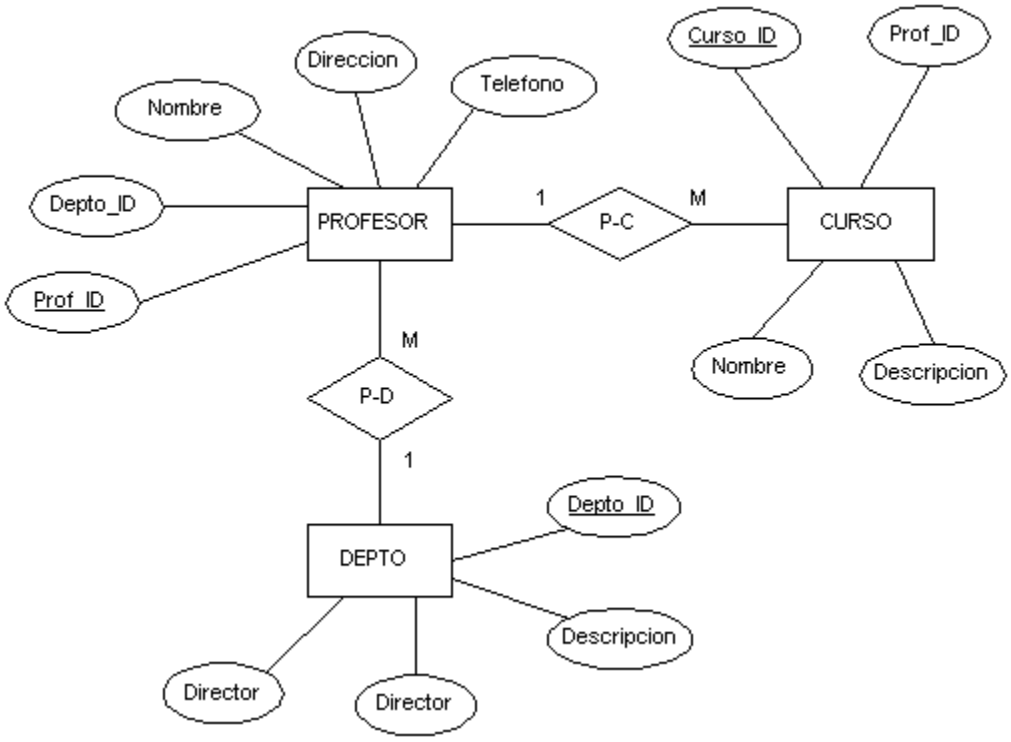


Fig. 18 Esquema Entidad-Relación

3.1.5. XML

XML, del inglés *eXtensible Markup Language* o Lenguaje de Marcado Extensible, procede del lenguaje SGML (*Standard Generalized Markup Language*, ISO 8879) desarrollado por un Grupo de Trabajo de XML formado bajo el auspicio del World Wide Web Consortium (W3C) en 1996 y dirigido por Jon Bosak de Sun Microsystems [46].

El lenguaje XML permite compartir datos entre diferentes equipos y aplicaciones de una manera segura, fiable y fácil. El objetivo de este lenguaje es definir un canal o medio común de comunicación entre sistemas [46] independientemente de su arquitectura.

Los documentos XML están hechos de unidades de almacenamiento llamadas etiquetas, las cuales contienen datos. Estos datos pueden estar hechos de caracteres y marcas. Las marcas codifican la descripción del esquema de almacenamiento y estructura lógica del documento.

```
<Books>
    <Book ISBN="0553212419">
        <title>Sherlock Holmes: Complete Novels...
        <author>Sir Arthur Conan Doyle</author>
    </Book>
    <Book ISBN="0743273567">
        <title>The Great Gatsby</title>
        <author>F. Scott Fitzgerald</author>
    </Book>
    <Book ISBN="0684826976">
        <title>Undaunted Courage</title>
        <author>Stephen E. Ambrose</author>
    </Book>
    <Book ISBN="0743203178">
        <title>Nothing Like It In the World</title>
        <author>Stephen E. Ambrose</author>
    </Book>
</Books>
```

Fig. 19 Ejemplo de documento XML

XML provee un mecanismo para imponer restricciones al esquema de almacenamiento y estructura lógica [46]:

- DTD (*Document Type Definition*) y XML schema: define la semántica del tipo de documento que se está creando (los elementos disponibles, las relaciones entre ellos, los atributos, posibles valores, etc.). Representa la estructura de una clase concreta del documento XML.
- XML Namespaces: proporcionan un método simple para distinguir los nombres de elementos y atributos que se utilizan en los documentos XML. El objetivo principal consiste en poder determinar que DTD debe considerar el analizador sintáctico del documento para analizar un elemento del mismo.

De acuerdo a las reglas que cumplen los documentos respecto a la especificación XML y a la definición de su estructura, se pueden distinguir estos dos tipos [46]:

- Bien formados: todos aquellos documentos que cumplen las especificaciones del lenguaje en lo que se refiere a las reglas sintácticas.
- Válidos: además de bien formados, responden a la estructura semántica definida por su DTD.

3.1.6. JSON

JSON, del inglés *JavaScript Object Notation*, está basado en un subconjunto del Lenguaje de Programación JavaScript [47]. Consiste en un formato informático ligero de intercambio de datos, basado en texto y legible, que sirve para representar objetos y otras estructuras de información. Se utiliza principalmente para transferir datos estructurados a través de una conexión de red, utilizando el proceso de serialización [48]. JSON está pensado principalmente para usarse como alternativa al uso de XML para transmitir información estructurada de forma asíncrona entre el servidor y los clientes.

Todo fichero JSON está constituido por dos estructuras: Una colección de pares nombre/valor y una lista ordenada de valores. En JSON, se presentan los siguientes componentes: objetos y arrays [48].

- Un objeto es un conjunto desordenado de pares nombre/valor.
- Un array está compuesto por un conjunto de objetos u otros arrays.

Un valor puede ser una cadena de caracteres con comillas dobles, un número, un valor booleano true/false, null, un objeto o un array. Estas estructuras pueden anidarse. Una cadena de caracteres es una colección de cero o más caracteres Unicode, encerrados entre comillas dobles, usando barras divisorias invertidas como escape. Un número es similar a un número C o Java, excepto que no se usan los formatos octales y hexadecimales.

JSON:

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

XML:

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Fig. 20 Ejemplo de JSON y su equivalente en XML

3.2. Plataforma Elgg

Elgg consiste en un framework de implementación de redes sociales. La plataforma consiste en un motor (en código libre) que proporciona la funcionalidad básica de toda red social [49]. Dicha funcionalidad comprende diversos módulos como el de login, registro, sistema de amigos, actividad de usuarios, gestión del sitio y administración, sistema de posts, comentarios, likes, etc. La plataforma provee de un sistema de añadido de funcionalidad mediante plugins. Este sistema permite incorporar nuevos requisitos de manera modular. Los plugins tienden a tratar con su propio contenido, lo que proporciona más estabilidad al sistema, evitando la desincronización de la información entre módulos [50]. El contenido creado por los diferentes plugins se puede mezclar de manera consistente con otras entidades o información de otros plugins, capturar eventos o registrar acciones.

Se desglosa a continuación una descripción de la estructura y modelos de datos de la plataforma Elgg basada en entidades.

3.2.1. Entidades

ElggEntity es la clase base para el modelo de datos de Elgg. ElggEntity es herencia de otras cuatro entidades principales, que proporcionan propiedades y métodos extra para manejar más fácilmente los diferentes tipos de datos [50].

- ElggObject: contenidos como blogs, cursos, archivos subidos y favoritos.
- ElggUser: un usuario del sistema.
- ElggSite: cada sitio Elgg dentro de una instalación.
- ElggGroup: sistemas de colaboración multi-usuario (llamado "Communities" en las versiones anteriores de Elgg)

La ventaja de este enfoque es que, aparte de poder modelar los datos con mayor facilidad, existe un conjunto de funciones siempre disponible para manejar los objetos, independientemente de su (sub)tipo [50]. Cada uno de ellos tiene sus propias características y atributos. Por ejemplo, los ElggObjects tienen un título y una descripción, los ElggUsers tienen un nombre de usuario y contraseña, etc... Sin embargo, debido a que todos heredan ElggEntity, todos tienen una serie de propiedades básicas y comunes:

- Un identificador numérico único global: **GUID**.
- Permisos de acceso. (Cuando un plugin solicita datos, nunca llega a tocar los datos que el usuario actual no tiene permiso para ver).
- Un subtipo no predefinido: pueden ser de cualquier forma única para describir un tipo particular de entidad. "Blog", "Foro", "Curso ", etc…
- Un propietario.
- El sitio que la entidad pertenece.
- Un contenedor, generalmente se usa para asociar el contenido de un grupo con el mismo.

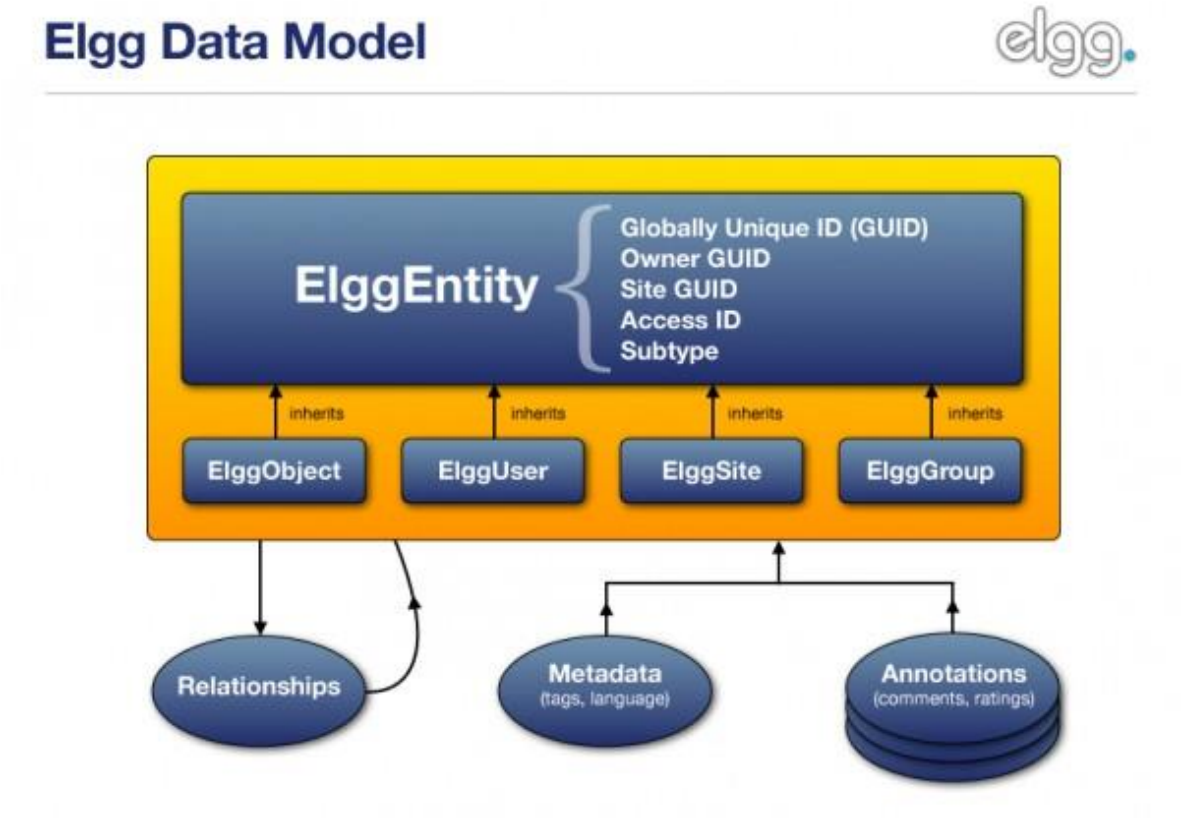


Fig. 21 Modelo de Datos básico de Elgg [50]

3.2.1.1. GUID

Un GUID es un número entero que identifica de forma exclusiva cada entidad en una instalación de Elgg. Se asigna automáticamente cuando la entidad primero se guarda y no se puede cambiar. Algunas funciones de la API de Elgg permiten trabajar con GUIDs en lugar de objetos ElggEntity.

3.2.1.2. ElggObject

El tipo de entidad ElggObject representa contenido arbitrario dentro de una instalación de Elgg: cosas como blogs, archivos cargados, etc… Además de las propiedades estándar de ElggEntity, los ElggObjects contienen los siguientes campos:

- Titulo
- Descripción

La mayoría de los otros datos sobre el objeto generalmente se almacenan a través de metadatos.

3.2.1.3. ElggUser

El tipo de entidad ElggUser representa a los usuarios dentro de una instalación de Elgg. Éstos son deshabilitados hasta que sus cuentas se han activado. Además de las propiedades estándar de ElggEntity, ElggUsers contienen los siguientes campos:

- Nombre
- Nombre de usuario
- Contraseña encriptada
- Email
- Fecha de la última acción ejecutada en el sistema
- Fecha del último login

3.2.1.4. ElggSite

El tipo de entidad ElggSite representa los sitios dentro de la instalación donde se ubica el engine de Elgg. La mayoría de las instalaciones tiene solamente una (es el caso del ABP). Además de las propiedades estándar de ElggEntity, ElggSite contiene los siguientes campos:

- El nombre del sitio
- Una descripción del sitio
- La dirección del sitio

3.2.1.5. ElggGroup

El tipo de entidad ElggGroup representa una asociación de usuarios de Elgg. Los usuarios pueden unirse o irse del grupo; además, el contenido del post puede ser restringido al grupo o ser público. Además de las propiedades estándar de ElggEntity, ElggGroup contiene los siguientes campos:

- El nombre del grupo
- Descripción del grupo

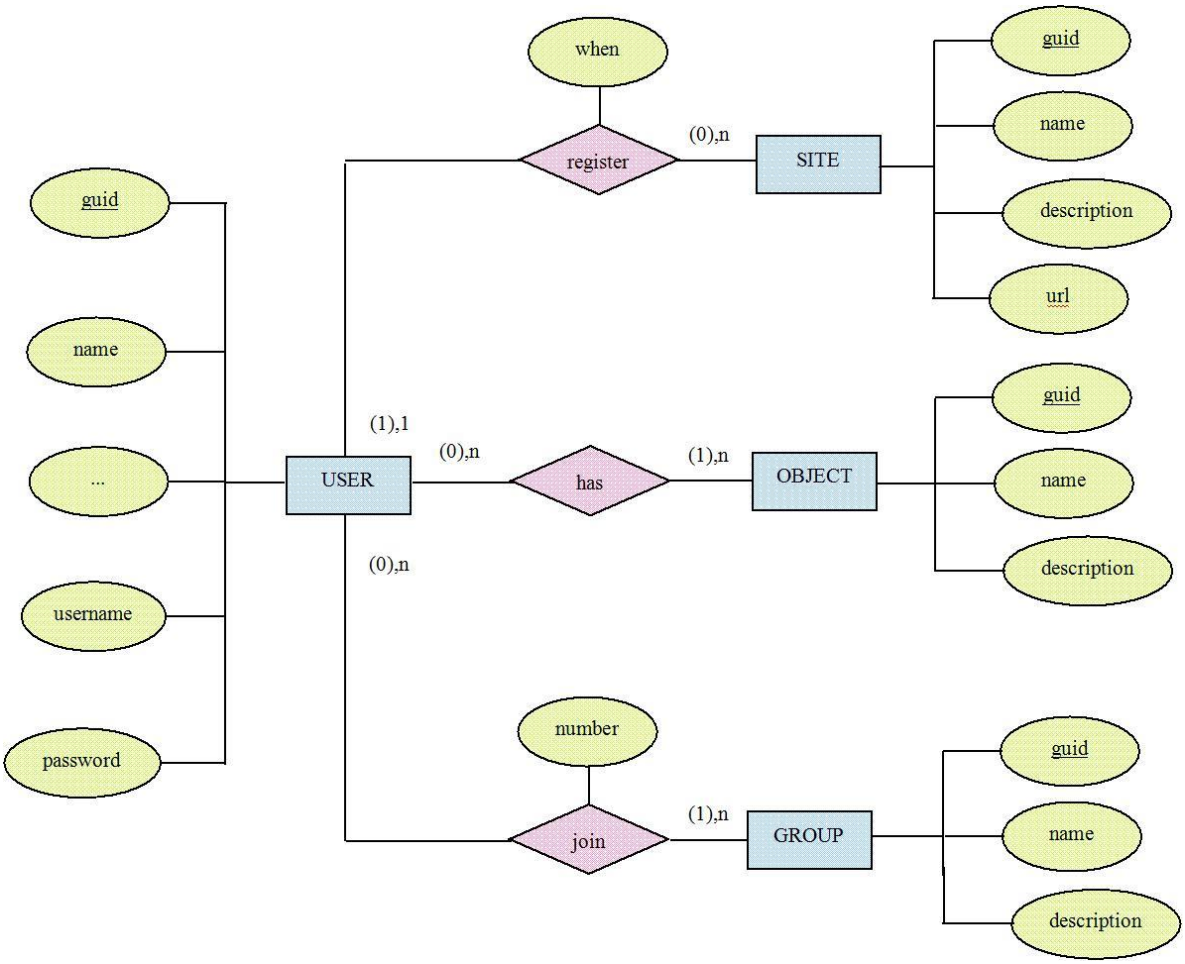


Fig. 22 Esquema entidad-relación reducido de Elgg

3.2.2. Propietario y Contenedores

Las entidades tienen un "**owner_guid"**, que define su propietario. Normalmente, esto se refiere al GUID de un usuario, aunque pueden existir entidades sin dueño (valor 0). El valor propietario de una entidad determina si se puede acceder o modificar dicha entidad.

Con el fin de facilitar la búsqueda de información dentro de un ámbito, el contenido en general tiende a ser "contenido" por el bien del usuario que lo envió, o el grupo al que pertenece el usuario que realiza la publicación. Esto significa que la propiedad "**container_guid"** del nuevo objeto se establecerá en el GUID de la entidad objetivo.

3.2.3. Anotaciones y Metadatos

Las anotaciones son piezas de datos adjuntos a una entidad donde los usuarios dejan información, opiniones, o un comentario relevante. Por ejemplo, un plugin de encuesta podría registrar votos

como anotaciones. Las anotaciones se almacenan como instancias de la clase **ElggAnnotation**. Cada anotación tiene:

- El tipo interno (comentario, valoración, like, etc…)
- Un valor (que puede ser una cadena o un entero)
- Un permiso de acceso distinto de la entidad al que está conectado
- Un propietario

Los metadatos consisten en información adicional y especifica de una entidad. Por lo general, los metadatos contienen la información concreta para un tipo de entidad. Por ejemplo, para un Curso de la clase ElggObject, sus metadatos comunes con otras entidades pueden ser la fecha de creación, el lenguaje, los tags, etc… y sus metadatos específicos pueden ser el número de secciones que tiene, sus permisos para el acceso y manejo de su contenido, etc…

3.3. Sistema Operativo Android

Android es un sistema operativo basado en el kernel de Linux diseñado principalmente para dispositivos móviles con pantalla táctil. Inicialmente desarrollado por Android Inc., Google respaldó económicamente su desarrollo y más tarde compró esta empresa en 2005 [51].

3.3.1. Componentes

Los componentes principales del sistema operativo de Android son [52]:

- **Aplicaciones**: Escritas en lenguaje Java, el SO contiene de base un cliente e-mail, SMS, calendario, mapas, navegador, contactos, etc...
- **Marco de trabajo de aplicaciones**: los desarrolladores tienen acceso completo a las mismas APIs del SO usado por las aplicaciones base. La arquitectura está diseñada para simplificar el uso y la reutilización de componentes; es decir, cualquier aplicación puede publicar su funcionalidad y cualquier otra aplicación puede luego hacer uso de las demás.
- **Bibliotecas**: Android incluye un conjunto de bibliotecas de C y C++ usadas por varios componentes del sistema (GPS, Cámara, etc…). Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android.
- **Runtime de Android**: Android incluye un set de bibliotecas que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia dentro de la máquina virtual Dalvik. Dicha máquina es la encargada de comunicar los procesos con el núcleo del dispositivo.
- **Núcleo Linux**: Android depende de un kernel Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

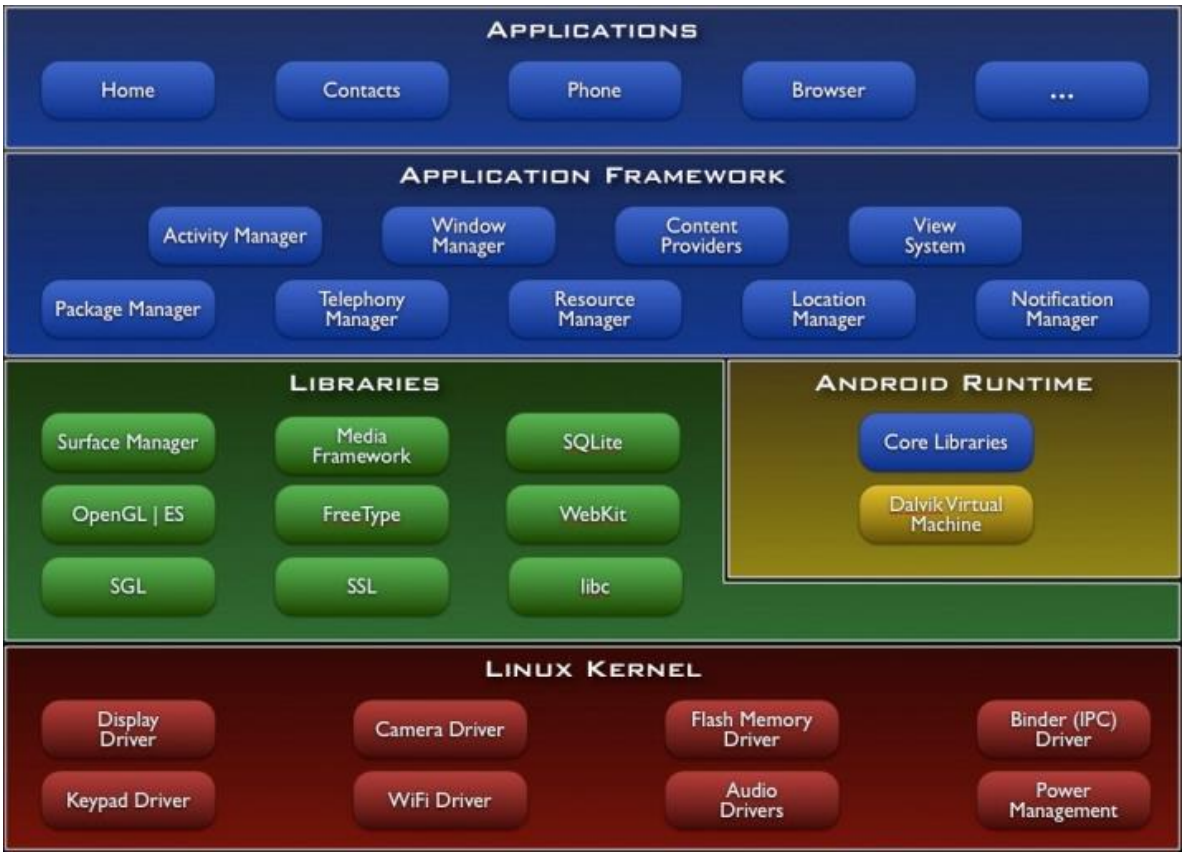


Fig. 23 Esquema de Arquitectura del SO Android

3.3.2. Versionado

El sistema operativo Android ha sido mejorado y optimizado en sucesivas versiones publicadas. Todas las versiones se identifican con el patrón "x.y.z" donde "x" identifica el nombre del SO (Kit-Kat, JellyBean, Froyo, etc…), "y" su versión y "z" su revisión [52]. El versionado supone una característica importante a la hora de diseñar cualquier aplicación en Android. En este caso, toda aplicación debe definirse su API mínima válida y su API de desarrollo. Debemos ser consecuentes a la hora de decidir las dos APIs, ya que dicha decisión influye directamente en la compatibilidad de los dispositivos que puedan reproducir la aplicación.

Dicho esto, podemos determinar la existencia de una relación compatibilidad/usabilidad. Desarrollar con una API mínima muy antigua limita la funcionalidad y los recursos disponibles a poder desarrollar mientras que la elección de una API mínima reciente posee mayores recursos, vistas, herramientas y optimizaciones que favorecen la usabilidad pero solo son soportadas por los dispositivos más nuevos.

A la hora de decidir que rango de acción deseamos que tenga nuestra aplicación, debemos considerar la relación compatibilidad/usabilidad. Además, debemos analizar la distribución de versiones dentro de la región donde queremos publicar nuestra aplicación. Esto último es muy

importante ya que sabiendo que versiones son las más distribuidas, podemos adaptar la aplicación al público y así asegurar su correcto acceso al mismo.

Version	Codename	API	Distribution
2.2	Froyo	8	0.8%
2.3.3 - 2.3.7	Gingerbread	10	14.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	12.3%
4.1.x	Jelly Bean	16	29.0%
4.2.x		17	19.1%
4.3		18	10.3%
4.4	KitKat	19	13.6%

Data collected during a 7-day period ending on June 4, 2014.
Any versions with less than 0.1% distribution are not shown.

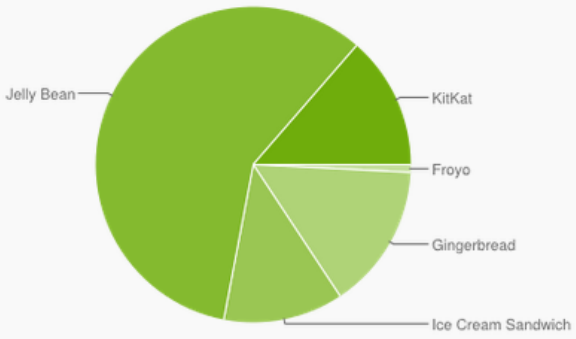


Fig. 24 Distribución mundial de versiones y API del SO Android en Junio de 2014 [28]

4. ANÁLISIS Y REQUISITOS

4.1. Introducción

En este apartado se presenta la Especificación de Requisitos Software (ERS) para el sistema "Africa Build App" y la capa de servicios web "Africabuild-Elgg-Rest-RCP-API". La estructura y formato de este apartado sigue el formato y directrices recomendadas para la especificación de Requisitos según el estándar IEEE 830-1998 [53].

4.1.1. Propósito

El objetivo de la Especificación de Requisitos Software es definir de manera clara y precisa todas las funcionalidades y restricciones que, en términos de requisitos software, presentarán los sistemas a construir. Además de definir todas las funcionalidades, características y restricciones que los sistemas deben contener, servirá para verificar y validar la corrección e integridad de los mismos.

4.1.2. Ámbito del sistema

El sistema "Africabuild-Elgg-Rest-RCP-API" se enmarca dentro de la plataforma Elgg del Africa Build Portal como un componente más de este y que tratará de ser una herramienta de comunicación con los usuarios finales a través de otras aplicaciones externas al portal. A su vez, el sistema "Africa Build App" se define como una de las aplicaciones externas mencionadas, la cual proporciona una alternativa más accesible y cómoda para los usuarios.

El primer sistema permite realizar consultas y acceso a los recursos de la plataforma Elgg de forma remota y a través de Internet. Será a manos del segundo sistema por el que el usuario podrá interactuar con la plataforma tomando como punto de entrada y muestra del contenido la App. El primer sistema actúa como puerta de acceso y "conversación" entre el portal y la App.

4.1.3. Definiciones y Acrónimos

Término	Definición
AFRICA BUILD	AFRICA BUILD consiste en un proyecto común entre distintas instituciones con el objetivo de apoyar y fomentar la salud, educación e investigación en países de África.
AFRICA BUILD Portal	AFRICA BUILD Portal es una red social de ámbito educativo donde se realizan y cumplen los requisitos asociados al proyecto AFRICA BUILD: fomentando la investigación y educación en países de África.
Servicio Web	Conjunto de aplicaciones y tecnologías que permite interoperar a través de internet.
e-learning	Sistema de educación a distancia completamente virtualizada a través de canales electrónicos, utilizando para ello herramientas o aplicaciones de hipertexto (correo electrónico, páginas web, foros de discusión, chats, etc…)
m-learning	Versión móvil del e-learning. Metodología de aprendizaje valiéndose del uso de dispositivos móviles tales como smartphones y tabletas entre otros con alguna forma de conectividad inalámbrica.
Plugin	Complemento software que añade una funcionalidad específica a una aplicación software existente.

Acrónimo	Definición
ERS	Especificación de Requisitos Software
IEEE	Institute of Electrical and Electronics Engineers
XML	eXtensible Markup Language
JSON	JavaScript Object Notation
PHP	PHP: Hypertext Preprocessor
REST	Representational State Transfer
API	Application Programming Interface
SO	Sistema Operativo
ABP	AFRICA BUILD Portal
APP	Aplicación móvil

4.2. Descripción general

Esta sección presenta una descripción de alto nivel del sistema con el fin de mejorar la comprensión de los requisitos.

4.2.1. Perspectiva del producto

Los servicios web forman parte de un sistema mayor, Elgg, mientras que la App consiste en una aplicación independiente desarrollado sobre la plataforma Android. Sin embargo, ambos desarrollos están influenciados por Elgg ya que para la comunicación y la interacción con el mismo dependemos del diseño y la estructura del portal, así como aprovechar las herramientas que nos proporcionan para poder recoger y enviar la información del mismo.

4.2.2. Funciones del producto

Los sistemas a desarrollar proporcionará la siguiente funcionalidad:

- Servicios Web:
  - Permitir el acceso a los recursos habilitados a los usuarios y administradores desde fuera de la plataforma.
  - Permitir aplicar extensiones del APB a través de nuevas aplicaciones y sistemas de terceros.
  - Proporcionar alternativas de comunicación e interacción con el portal para los usuarios.
- App:
  - Acceder a través de los servicios web al ABP mediante dispositivos móviles conectados a Internet.
  - Proporcionar la funcionalidad ya existente en el portal como aplicación. Dicha funcionalidad está asociada al ámbito de los cursos que proporciona el ABP, su contenido, recursos de aprendizaje y actividad.
  - Permitir la interacción con el portal a través de una alternativa más usable para Smartphones y Tablets.

4.2.3. Características de los usuarios

El sistema únicamente distingue un tipo de usuario el cual se centra en su mayoría al usuario final para ambos sistemas. El enfoque de los Servicios web y la App se basa principalmente en la obtención de información y funcionalidad común del portal para todos los usuarios. Dicho esto, ambos sistemas están enfocados para la interacción con el mismo de cara al uso del portal y no para su gestión, control y/o administración.

4.2.4. Restricciones

Con respecto a los servicios web, el servidor debe incorporar una versión de PHP 5.0 o superior para su correcto funcionamiento, además de incorporar la plataforma Elgg y habilitar el acceso a la librería de servicios web básica que ofrece Elgg, deshabilitado por defecto por cuestiones de seguridad.

En lo que respecta a la App, su instalación en dispositivos móviles y tablets está sujeto a versiones del Sistema Operativo Android igual o mayores a la versión 2.2, correspondiente a la API 8 de dicho SO. El dispositivo móvil con Android instalado deberá tener acceso a Internet de Banda ancha mínimo de 2.5G (114 kbit/s) o WIFI para asegurar una correcta velocidad de conexión, reacción del sistema y evitar pérdida de información.

4.2.5. Suposiciones y dependencias

Se supone que el futuro diseñador y/o usuario de los servicios web tiene conocimiento del uso de las tecnologías relacionadas con la comunicación a los mismos. Con respecto a la App, se supone que el usuario tiene nociones de conocimiento sobre uso de Apps, dispositivos móviles, smartphones y tablets.

En un ámbito más global, se manifiesta la clara dependencia de los sistemas a desarrollar con el ABP, dejando a la vista la suposición de que todo usuario que utilice estas soluciones conoce el portal, es un usuario valido y registrado en el mismo.

4.3. Requisitos específicos – Servicios Web

Las dos siguientes secciones contienen los requisitos a un nivel de detalle suficiente como para permitirnos diseñar un sistema que satisfaga los mismos, de tal manera que nos permita planificar y realizar tanto el desarrollo como sus pruebas.

4.3.1. Interfaces externas

- Interfaz Hardware
    - El sistema debe funcionar en distintas plataformas (Intel, AMD, ARM, etc…) y arquitecturas hardware (32 y 64 bits).
    - El sistema debe funcionar para la versión de Elgg implementada actualmente (1.8).
    - El sistema se basa en PHP para la implementación de los Servicios Web.
- Interfaz Software
    - El sistema debe funcionar en el SO en el que esté ubicado Elgg.
    - El sistema debe poder enviar información a otros sistemas independientemente de su plataforma siempre y cuando pueda comunicarse a través del protocolo HTTP.
- Interfaz de comunicación

- El sistema debe estar basado en Servicios Web, compatible tanto para JSON como XML y ser introducido dentro del módulo de servicios web y comunicación de la plataforma Elgg para su uso.

4.3.2. Funciones

- Consulta de información
  - El sistema reconoce el Servicio Web al que va dirigida la consulta.
  - El sistema debe comprobar la corrección de los parámetros y proporcionar por defecto aquellos que puedan faltar y sean necesarios para la solicitud.
  - El sistema debe extraer la información en un mismo esquema para una misma función o consulta.
  - El sistema debe comprobar la validez y los permisos de usuario para la realización de la consulta.
  - El sistema debe devolver la información en lenguaje XML o JSON.
- Entrega y actualización de información
  - El sistema debe notificar adecuadamente cuando una modificación, actualización o entrega de datos ha sido satisfactoria o no.
  - Los datos a entregar deben ser enviados en HTTP a través de la Url o contenido en XML o JSON.
  - El sistema no debe realizar ninguna modificación si los parámetros no son los adecuados.
  - El sistema debe respetar y ser acorde al sistema de permisos y acceso de la plataforma Elgg donde está instalado.
- Consulta de documentación
  - El sistema debe permitir a través de una libre petición generar una correcta pero breve documentación de los servicios y funcionalidad disponible.

4.3.3. Requisitos de rendimiento

- El sistema debe poder ser accesible simultáneamente, proporcionando el mismo servicio para cada solicitud.

4.3.4. Restricciones de diseño

- Los servicios deben poseer la estructura de Servicios web de Elgg.
- Los servicios debe utilizar el plugin de Moodle-Manager desarrollado en el ABP para la obtención de la información asociada a cursos y contenido educativo.
- Con respecto a los permisos de acceso, se debe utilizar la funcionalidad y mecánicas existentes en Elgg dedicadas a Servicios web.

4.3.5. Atributos del sistema

- **Fiabilidad**: El sistema será fiable siempre y cuando la solicitud de información se realice con los datos adecuados. Un parámetro erróneo en la solicitud proporcionará bien una excepción (si no existe) o bien información equivocada (si existe). En el caso de omisión de parámetros, el sistema asume los valores por defecto de las consultas.

- **Mantenibilidad**: El desarrollo del sistema deberá basarse en estándares y convenciones ampliamente establecidas, de forma que el desarrollo futuro del mismo no se vea condicionado o dificultado por el uso de técnicas y/o tecnologías propietarias y pueda ser continuado partiendo de la solución aquí propuesta. Esta estandarización debe aplicarse a todas las facetas del sistema: arquitectura, diseño, implementación y documentación.
- **Flexibilidad**: El sistema deberá ser flexible y admitir la posibilidad de incorporar nuevos servicios o de modificación de los actuales sin comprometer el funcionamiento del resto del sistema.
- **Seguridad**: El software permitirá comprobar si el usuario asociado al token enviado en la solicitud dispone de los permisos pertinentes sobre cada Servicio Web. Será competencia de la plataforma Elgg comprobar la legitimidad del token. El software realizará consultas de lectura o escritura sobre el ABP, comprobando previamente los permisos y acceso disponibles para el usuario.

4.4. Requisitos específicos – App

4.4.1. Interfaces externas

- Interfaz Hardware
  - El sistema debe funcionar en todas las plataformas (Intel, AMD, ARM, etc…) y arquitecturas hardware (x32 y x64 bits) que puedan soportar el Kernel del SO de Android.
  - El sistema se basa en Java para la implementación de la lógica interna y en XML para la definición de las vistas y layouts.
- Interfaz Software
  - El sistema debe funcionar en el SO de Android.
  - El sistema debe poder enviar información a otros sistemas independientemente de su plataforma siempre y cuando pueda comunicarse a través del protocolo HTTP.
  - El sistema debe ser accesible para su descarga desde la tienda software de Google Play de manera gratuita.
- Interfaz de comunicación
  - El sistema contiene un módulo que debe estar basado en Servicios Web en JSON para su comunicación con el ABP a través de la capa de servicios web.

4.4.2. Funciones

- Consulta de información
  - El sistema reconoce e identifica al usuario registrado en el ABP.
  - El sistema debe mostrar de manera análoga al portal el contenido asociado a cursos, actividad y recursos de aprendizaje asociados a los mismos al usuario.
  - El sistema debe actualizar la información y tener una sincronía con el portal.
  - El sistema debe respetar los permisos del usuario según los mismos que proporciona el portal.
  - El sistema debe devolver la información y mostrar el contenido mínimo en inglés, siendo posible su migración a otros idiomas.

  - El sistema debe mostrar la misma información para cualquier dispositivo diferente que pueda cargar la aplicación.
- Entrega y actualización de información
  - El sistema debe ver reflejado adecuadamente cuando una modificación, actualización o entrega de datos ha sido satisfactoria o no.
  - Los datos a enviar deben ser enviados en HTTP a través de la Url o contenido en XML o JSON.
  - El sistema no debe realizar ninguna modificación si los parámetros no son los adecuados.
  - El sistema debe respetar y ser acorde al sistema de permisos y acceso de la plataforma Elgg donde está instalado.

4.4.3. Requisitos de rendimiento

- El sistema debe poder ser accesible a través de la banda ancha o consumo de datos del dispositivo anfitrión para una tecnología de conexión 2.5G o mayor.
- El sistema debe bloquear la interacción del usuario durante la carga de los datos para evitar problemas y errores en el contenido.
- El sistema debe ahorrar en la medida de lo posible el consumo de batería y CPU del dispositivo.

4.4.4. Restricciones de diseño

- El sistema debe ser implementado según la arquitectura de desarrollo de Apps para Android.
- Dicha arquitectura debe incluir las librerías necesarias para el acceso y comunicación con servicios web.
- El sistema debe seguir las pautas de diseño, visualización y estilos del ABP.
- El sistema debe poder redirigir los recursos basados en URLs al navegador web que proporcione el dispositivo si existe.
- El sistema debe ser completamente usable para cualquier dispositivo que cumpla el requisito de versión instalada de Android 2.2 o mayor (API 8).

4.4.5. Atributos del sistema

- **Fiabilidad**: El sistema será fiable siempre y cuando la solicitud de información se realice con los datos adecuados. Un error en el sistema proporcionará bien una excepción y la salida de la aplicación para no comprometer el dispositivo con el objetivo de que el usuario no debe perder el control del mismo.
- **Mantenibilidad**: El desarrollo del sistema deberá basarse en estándares y convenciones ampliamente establecidas, de forma que el desarrollo futuro del mismo no se vea condicionado o dificultado por el uso de técnicas y/o tecnologías del desarrollador y pueda ser continuado partiendo de la solución aquí propuesta. Se proporcionará comentarios y explicaciones de procesos dentro de la implementación para favorecer la comprensión del mismo.
- **Portabilidad**. El sistema desarrollado deberá ser independiente del dispositivo de ejecución siempre y cuando tenga el SO de Android.

- **Flexibilidad**: El sistema deberá ser flexible y admitir la posibilidad de incorporar nueva funcionalidad sin comprometer el resto del sistema o su visualización ya implementada.
- **Seguridad**: El software permitirá comprobar la validez del usuario a través de un login directamente asociado a la autenticación proporcionada por los servicios web. Será competencia de la plataforma Elgg realizar la validación del usuario y, junto con los servicios web, el tratamiento de las consultas.

5. DISEÑO E IMPLEMENTACIÓN

En el desarrollo de este apartado se ha optado por desglosar el diseño y de la implementación de cada una de las soluciones por separado. Sin embargo, ambas determinan su funcionalidad mediante la definición de casos de uso. Para el caso de la implementación, se define en el plugin la documentación de la API desglosada en funciones mientras que en la App se argumenta la implementación de la interacción e interfaces y vistas.

5.1. Plugin ELGG-AFRICABUILD-REST-RCP-API

Elgg permite la ampliación de su funcionalidad a través de plugins y la solución debe ser implementada como tal, respetando el diseño, estructura, implicaciones, etc… dentro del ABP. Además, el acceso y manipulación de los datos debe ser tratada a través de las librerías y funciones proporcionadas por el engine de Elgg.

5.1.1. Diseño

En este apartado se describe todo el proceso de diseño del plugin. En él se describen las pautas, ámbito, agentes y demás consideraciones para su posterior implementación. Para el diseño del plugin se han ido considerando los casos de uso asociados al usuario dentro del portal. La API debe proporcionar toda la funcionalidad que pueda necesitar el usuario. Dicha funcionalidad debe respetar en todo momento las restricciones del ABP.

5.1.1.1. Actores

La API solo estará disponible para **sistemas en modo usuario**. Definimos **modo usuario** a la modalidad de funcionalidad común a todo usuario registrado en el ABP. Definimos usuario a todo miembro del ABP cuyo objetivo es realizar las actividades, participar en la comunidad de los cursos y grupos en los que esté inscrito y acceder al contenido del portal.

Por motivos de seguridad, los administradores del ABP solo pueden usar la API en modo usuario. La administración del portal así como la potestad de habilitar o deshabilitar la API recae en el administrador. Este solo puede acceder a la gestión del portal a través de la versión web.

El otro actor a considerar es el **Engine de Elgg del ABP**. El objetivo del engine consiste en proporcionar la información que necesite el sistema en modo usuario. El ABP se encargará de la lógica interna de almacenamiento de la información. Será labor de la API solicitar y recoger la información, formatearla y enviar vía HTTP la respuesta.

5.1.1.2. Resumen del ámbito

El plugin integrado dentro del ABP deberá tener acceso a toda la información a la que pueda acceder el usuario. Esta información consiste en los siguientes tipos de datos:

- Entidades visibles (Cursos, Blogs, Bookmarks, Ficheros, Grupos, Debates y Páginas).
- Comunidad del ABP (Usuarios, anotar entidad, visualizar perfil y actividad).
- Mensajería (Enviar mensajes y buzones de entrada y enviados).

Por otro lado, la API debe proporcionar la funcionalidad asociada al módulo de "e-learning" del portal. El módulo comprende el acceso a los cursos y su contenido:

- Matriculación a un curso
- Visualizar contenido (Video, Ficheros y Webcasts).
- Responder quizzes y ver resultados.

Se descarta de la funcionalidad de la API el registro de usuarios. Dicho registro está sujeto a un control de acceso (Anti-spam y reCaptcha) que no puede ser simulado en la API por cuestiones de tiempo, seguridad e implementación.

5.1.1.3. Diagrama de casos de uso

Para determinar la funcionalidad de la API, se ha diseñado su diagrama de casos de uso. En el diagrama se muestran todos los casos de uso que comprende el plugin, indicando la interacción entre actores. Para una mejor comprensión del diagrama, se ha dividido en dos partes: el contenido general del ABP (Fig. 26) y el módulo de "e-learning" asociado a los cursos (Fig. 25).

Más tarde se describen uno a uno cada caso de uso. Para la definición de cada caso de uso se utilizará una descripción en alto nivel, proporcionando una visión general del mismo, precondiciones, postcondiciones y respuesta obtenida.

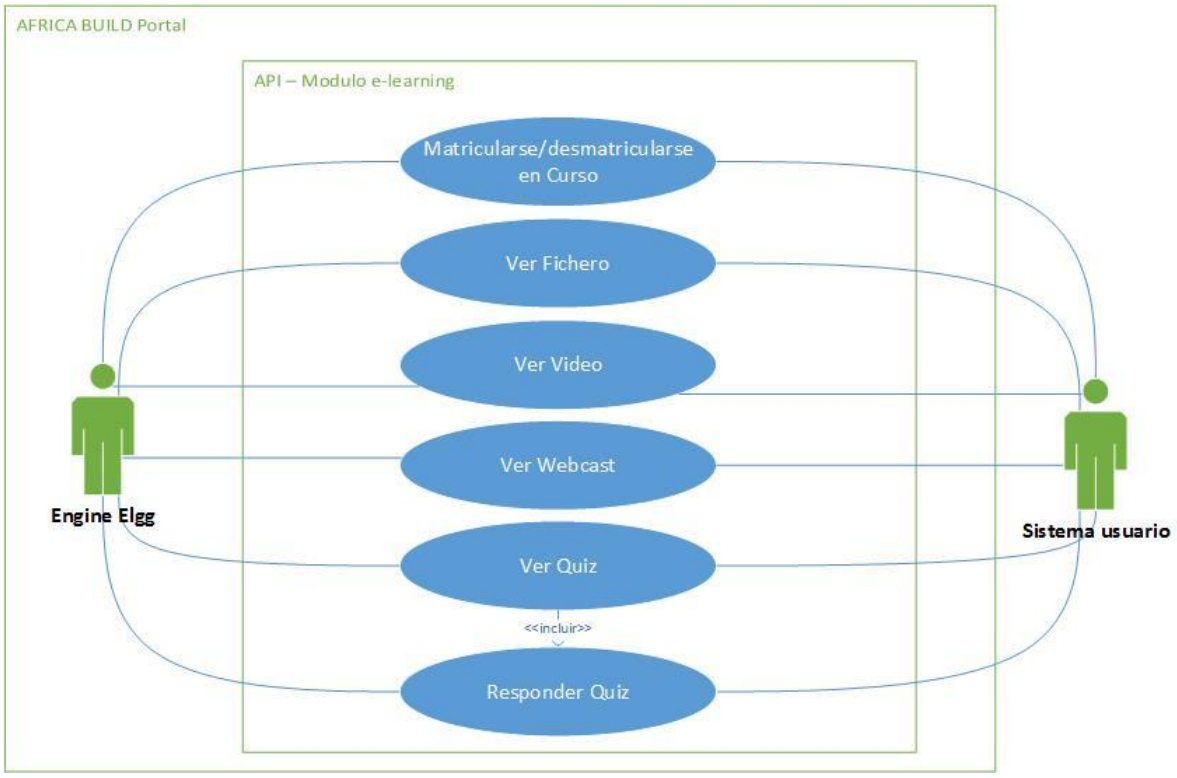


Fig. 25 Diagrama de casos de uso: API - Modulo e-learning

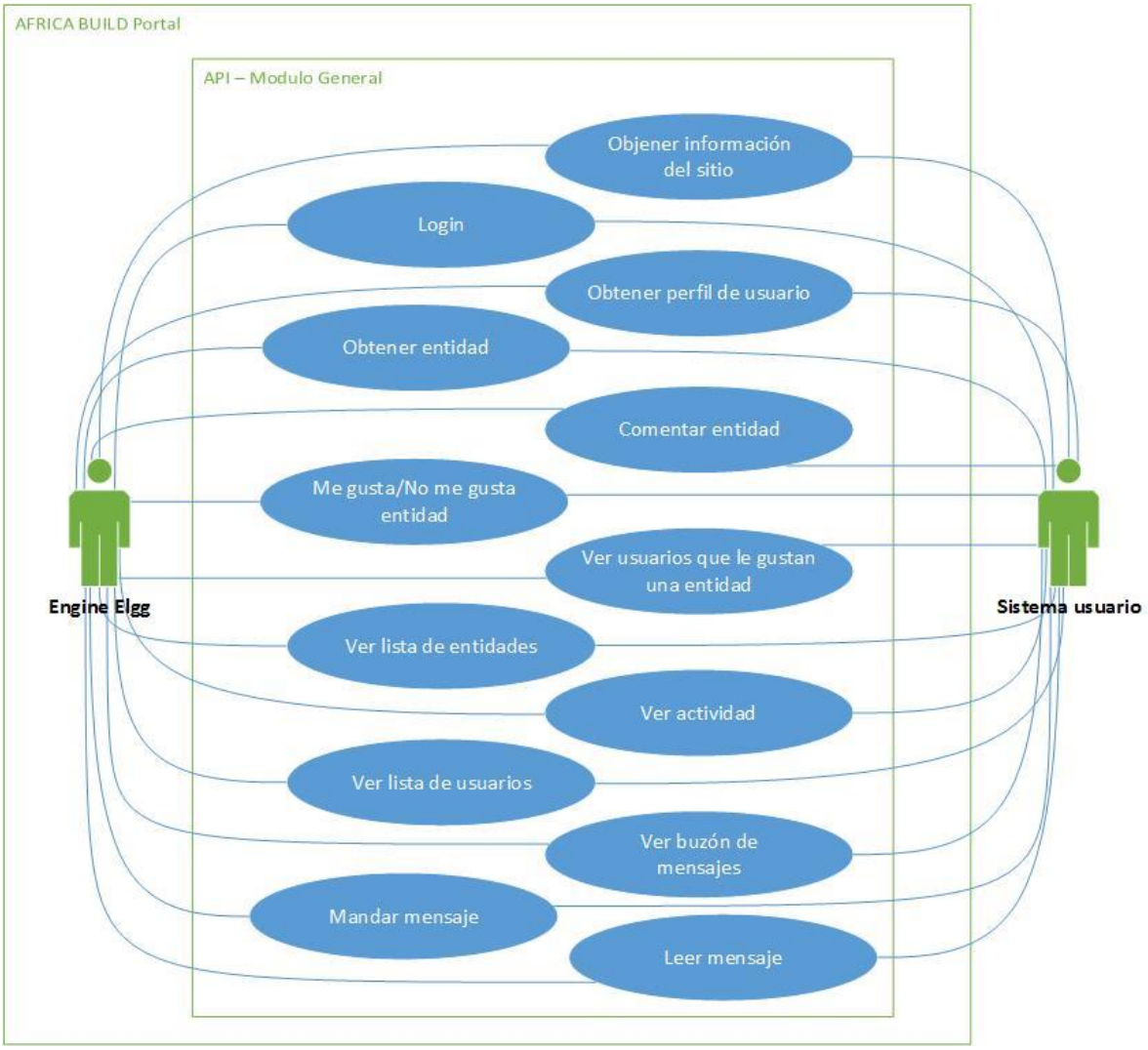


Fig. 26 Diagrama de casos de uso: API - Módulo General

5.1.1.4. Casos de Uso

Se enumeran y describen a continuación todos los casos de uso definidos en el diagrama en un formato más extendido. Los casos de uso son definidos en alto nivel y marcan la pauta del comportamiento del sistema frente a las peticiones del usuario. Para todos los casos de uso sus actores son el engine del ABP y el sistema en modo usuario.

I. Caso de uso **"Obtener información del sitio"**

- Propósito: Obtener información general sobre el sitio donde se ubica el ABP.
- Precondición: -
- Postcondición: -
- Datos:

- Entrada: -
- Respuesta: Información del sitio.

II. Caso de uso **"Login"**

- Propósito: Valida la autenticación de un usuario en el ABP.
- Precondición: Usuario registrado en el ABP.
- Postcondición: Usuario logueado en el sistema.
- Datos:
  - Entrada: Nombre de usuario y contraseña.
  - Respuesta: OK si éxito. Error en caso contrario.

III. Caso de uso **"Obtener perfil de usuario"**

- Propósito: Permite obtener información general del usuario.
- Precondición: Usuario logueado en el ABP.
- Postcondición: -
- Datos:
  - Entrada: -
  - Respuesta: Datos de perfil del usuario.

IV. Caso de uso **"Obtener entidad"**

- Propósito: La función devuelve la información completa de una entidad. Dicha información consiste en los campos genéricos a una entidad, el identificador de la entidad contenedora si lo tiene y sus metadatos.
- Precondición: Usuario logueado en el ABP.
- Postcondición: -
- Datos:
  - Entrada: Identificador de entidad.
  - Respuesta: Entidad completa con meta-información.

V. Caso de uso **"Comentar entidad"**

- Propósito: Añade una nueva anotación de tipo comentario a la entidad. El comentario tiene asociado el identificador del usuario que lo realiza.
- Precondición: Usuario logueado en el ABP. Entidad visible por el usuario.
- Postcondición: Entidad anotada con un comentario. Actualización de la actividad asociada a nuevo comentario.
- Datos:
  - Entrada: Comentario
  - Respuesta: OK si éxito. Error en caso contrario.

VI. Caso de uso **"Me gusta/no me gusta entidad"**

- Propósito: Añade o borra la anotación de tipo "like" asociado a una entidad. La anotación tiene asociado el identificador del usuario que lo realiza
- Precondición: Usuario logueado en el ABP. Entidad visible por el usuario.

- Postcondición: Entidad anotada con un me gusta/ Borrado de la anotación me gusta de la entidad. Actualización de la actividad asociada a la anotación si le gusta al usuario.
- Datos:
  - Entrada: Me gusta/no me gusta.
  - Respuesta: OK si éxito. Error en caso contrario.

VII. Caso de uso **"Ver usuarios que le gustan una entidad"**

- Propósito: Devuelve una lista de usuarios que tienen asociada la anotación "like" en una entidad, dado su identificador.
- Precondición: Usuario logueado en el ABP. Entidad visible por el usuario.
- Postcondición: -
- Datos:
  - Entrada: Identificador de entidad.
  - Respuesta: Lista de usuarios que han anotado "me gusta" en la entidad.

VIII. Caso de uso **"Ver lista de entidades"**

- Propósito: Devuelve una lista de entidades. La lista puede filtrarse por su tipo o por entidad contenedora. De esta manera, podemos obtener entidades especificas a otra (secciones de un curso, recursos de una sección, etc…)
- Precondición: Usuario logueado en el ABP.
- Postcondición: -
- Datos:
  - Entrada: Identificador de entidad contenedora (opcional), tipo de entidad (opcional), tamaño de la lista e índice de búsqueda.
  - Respuesta: Lista de entidades visibles por el usuario y coincidentes con los parámetros de entrada.

IX. Caso de uso **"Ver actividad"**

- Propósito: Lista la actividad asociada a una entidad (o al portal si la entidad está definida). La lista esta ordenada por fecha de más reciente a más antiguo.
- Precondición: Usuario logueado en el ABP. Actividad asociada a una entidad visible por el usuario.
- Postcondición: -
- Datos:
  - Entrada: Identificador de entidad contenedora (opcional), tamaño de la lista e índice de búsqueda.
  - Respuesta: Lista de actividad visible por el usuario y coincidente con los parámetros de entrada.

X. Caso de uso **"Ver lista de usuarios"**

- Propósito: Proporciona una lista de usuarios. Si se proporciona en la entrada el identificador de una entidad de tipo curso, la lista devuelve los usuarios matriculados a dicho curso.
- Precondición: Usuario logueado en el ABP.

- Postcondición: -
- Datos:
  - Entrada: Identificador de entidad curso (opcional), tamaño de la lista e índice de búsqueda.
  - Respuesta: Lista de usuarios coincidente con los parámetros de entrada. Si existe el identificador de entidad y corresponde a un curso devuelve la lista de usuarios matriculados.

XI. Caso de uso **"Ver buzón de mensajes"**

- Propósito: Lista los mensajes en la bandeja de entrada leídos y no leídos.
- Precondición: Usuario logueado en el ABP.
- Postcondición: -
- Datos:
  - Entrada: -
  - Respuesta: - Lista de mensajes sin leer y leídos

XII. Caso de uso **"Mandar mensaje"**

- Propósito: Envía un mensaje a un usuario del ABP.
- Precondición: Usuario logueado en el ABP.
- Postcondición: Mensaje enviado y pendiente de lectura en el buzón del usuario receptor.
- Datos:
  - Entrada: Mensaje y receptor.
  - Respuesta: OK si éxito. Error en caso contrario.

XIII. Caso de uso **"Leer mensaje"**

- Propósito: Devuelve el contenido de un mensaje y se marca como leído.
- Precondición: Usuario logueado en el ABP.
- Postcondición: Mensaje leído.
- Datos:
  - Entrada: Identificador del mensaje.
  - Respuesta: Contenido del mensaje.

XIV. Caso de uso **"Matricularse/desmatricularse en un Curso"**

- Propósito: Añade o elimina una relación de matriculación entre la entidad curso y el usuario. En el caso de estar matriculado, el usuario podrá acceder a los recursos de aprendizaje del curso.
- Precondición: Usuario logueado en el ABP.
- Postcondición: Modificación de la relación matriculado entre la entidad curso y el usuario. Acceso modificado del usuario a las entidades contenidas dentro del curso. Actualización de la actividad asociada al curso si el usuario se matricula.
- Datos:
  - Entrada: - Identificador de entidad curso.
  - Respuesta: OK si éxito. Error en caso contrario.

XV. Caso de uso **"Ver fichero"**

- Propósito: Devuelve la URL de un fichero. La URL se define a través de la URI del fichero dentro del sistema y su visualización a través del visor de ficheros de Google (Google Docs Viewer).
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso contenedor del fichero.
- Postcondición: -
- Datos:
  - Entrada: Identificador de recurso fichero.
  - Respuesta: URL de acceso al fichero.

XVI. Caso de uso **"Ver video"**

- Propósito: Devuelve la URL de un video. La URL está asociada a un enlace de video en YouTube.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso contenedor del video.
- Postcondición: -
- Datos:
  - Entrada: Identificador de recurso video.
  - Respuesta: URL de acceso al video.

XVII. Caso de uso **"Ver webcast"**

- Propósito: Devuelve la URL asociada a un webcast. La URL está asociada a los webcast denominados DUDAL, de la página web www.raft.unige.ch donde están contenidos las conferencias, docencias y presentaciones de los cursos del ABP.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso contenedor del webcast.
- Postcondición: -
- Datos:
  - Entrada: Identificador de recurso webcast.
  - Respuesta: URL de acceso al webcast.

XVIII. Caso de uso **"Ver quiz"**

- Propósito: Devuelve los datos necesarios para completar un quiz: preguntas, respuestas posibles, distribución, tiempo límite y número de intentos realizados. Los tipos de preguntas son: multiple elección, elección simple, unir, calcular, completar, respuesta corta y desarrollar.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso contenedor del quiz.
- Postcondición: Quiz asociado al usuario. Inicio del cronometro si existe tiempo limite
- Datos:
  - Entrada: Identificador de recurso quiz.

- Respuesta: Contenido del quiz: preguntas, tipo de pregunta y opciones.

XIX. Caso de uso **"Responder quiz"**

- Propósito: Devuelve la corrección del quiz entregado, dicha corrección implica solo los tipos de preguntas que pueden ser corregidas en el sistema (tipo desarrollo descartado). Se devuelve junto a la corrección las respuestas correctas a las preguntas fallidas y la puntuación del quiz.
- Precondición: Usuario logueado en el ABP. Quiz completado.
- Postcondición: Quiz asociado al usuario resuelto y corregido.
- Datos:
  - Entrada: Identificador de recurso quiz y respuestas al quiz.
  - Respuesta: Corrección y puntuación del quiz.

Una vez descritos los casos de uso, se dispone a realizar la implementación que responda a los mismos y cumpliendo los requisitos asignados a la solución.

5.1.2. Implementación

El desarrollo de la funcionalidad de la API se ha realizado en PHP siguiendo el esquema de arquitectura de plugins para Elgg (ver Anexo A). La implementación de la API debe ser lo más intuitiva posible; es decir, el usuario debe intuir a priori el uso de las funciones a través de sus definiciones, y parámetros de entrada. En este apartado no se adjunta código por motivos de propiedad intelectual.

5.1.2.1. Bloques

Se definen un conjunto de bloques o secciones dentro de la API. Estos bloques ayudan en la comprensión de la funcionalidad, al mantenimiento en futuras ampliaciones de la API:

- **Funciones Auxiliares**: Se agrupan aquí todas las funciones de carácter general para el correcto uso del resto de la API. Se añaden también las funciones que proporciona el engine de Elgg por defecto: Autenticación y lista de funciones de la API.
- **Entidades**: El portal encapsula toda la información en entidades, ya sea en la propia entidad o en forma de metadatos y/o anotaciones. Se puede tratar entonces de manera estandarizada toda la información a la que puede acceder un usuario. Aquí se agrupan todas las funciones que trabajan con entidades y su contenido.
- **Listas**: El bloque contiene todas las funciones que devuelven listas de contenido: Entidades, usuarios y actividad.
- **Recursos de Aprendizaje**: Aquí se encapsula todas las funciones que interactúan con los recursos de aprendizaje del portal. Dichos recursos siempre están asociados a una entidad de tipo curso.
- **Mensajes**: En este bloque se ubican las funciones necesarias para el uso del sistema de mensajería que proporciona el ABP.

Cada bloque se funciones corresponderá a un fichero PHP dentro de la carpeta "/lib" del plugin (Ver Anexo A): auxiliar.php, entities.php, lists.php, elearning.php y messages.php.

5.1.2.2. Relación entre bloques: Entidades y Listas

La API debe proporcionar un sistema de identificación de la información. Esto es muy importante ya que las funciones deben proporcionar los datos necesarios para que el sistema usuario pueda acceder a la información mediante un recorrido en profundidad. Por ejemplo, si el sistema usuario desea información sobre los cursos disponibles, este puede solicitar una lista de dichos cursos; pero si ahora desea más información de un curso en concreto, la lista debe proporcionarle el identificador del curso que quiere para hacer la nueva petición. A su vez, ese curso le puede proporcionar una lista de entidades contenidas (Secciones, por ejemplo) y así profundizar más en el contenido.

Se produce así una relación entre la funcionalidad de los bloques de entidades y listas. La idea consiste en permitir al sistema usuario una iteración entre entidades contenidas en otras. El engine se encarga de devolver las entidades que sean visibles por el usuario.

Para la iteración entre entidades y listas utilizaremos el atributo **GUID** de la entidad y el **GUID contenedor** de la misma. Dicho atributo identifica inequívocamente cualquier entidad definida dentro del ABP. De esta forma, junto con el concepto de entidades contenedoras, podemos realizar dicha iteración (Fig. 27). Además, con este sistema podemos realizar el recorrido en ambos sentidos si la entidad proporciona dentro de su información la entidad que lo contiene.

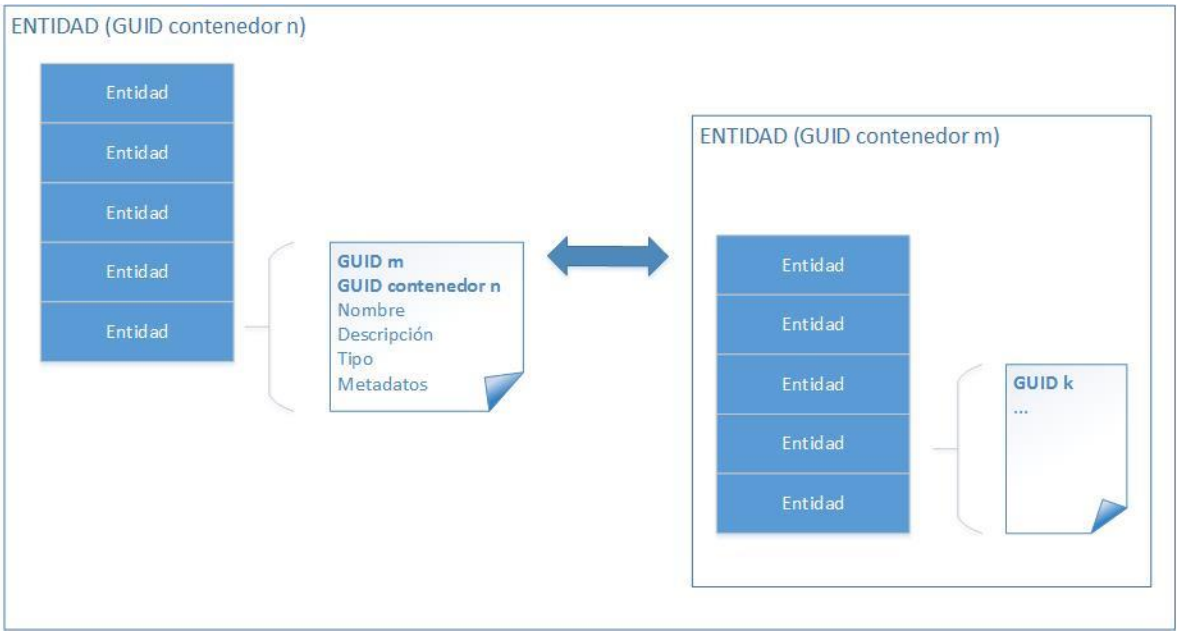


Fig. 27 Lógica de Iteración entre listas y entidades

En definitiva, los servicios web a implementar deben seguir la estructura y relaciones de las entidades definidas en Elgg, con el objetivo de que el producto a diseñar sea lo más coherente e intuitivo posible, además de facilitar su desarrollo y mantenimiento.

5.1.2.3. Definición de URL

Toda función expuesta siguiendo la arquitectura de servicios web del engine de Elgg es de la forma:

http://***{sitio}***/elgg/services/api/rest/***{xml|json}***/?method=***{function}[&param=value]***

Donde "sitio" es el host del ABP, "xml|json" es el formato de entrada y salida de datos, "function" el nombre de la función expuesta y "params" el conjunto de pares atributo-valor para las funciones que lo requieran.

5.1.2.4. Formato de respuesta

Elgg proporciona una estructura básica de formato de respuesta a las peticiones tanto en JSON como XML. Dicho formato tiene como cabecera principal la etiqueta *elgg* con los campos *status* y *result*. El primero proporciona información sobre si la petición se ha realizado satisfactoriamente, indicándose así con el valor 0. El segundo contiene la respuesta de la consulta cuya estructura define el desarrollador de la API.

```
▼<elgg>
   <status type="integer">0</status>
   <result type="string">Hi . I'm working</result>
 </elgg>
```

Fig. 28 Ejemplo de respuesta XML en Elgg

5.1.2.5. Funcionalidad de la API

Finalmente, correspondiendo al diseño, arquitectura, lógica y formatos a utilizar, se ha desarrollado la API con la siguiente funcionalidad. El apartado bloque define la sección a la que pertenece la funcionalidad, el apartado función determina el tipo de función HTTP asociado (GET o POST) y el apartado método define en nombre de la función expuesta en la API (parámetro "method" en la URL).

Bloque	Función	Método
Funciones Auxiliares	GET	abp_aux_hello
	GET	abp_aux_site
	GET	system.api.list
	POST	auth.gettoken
	GET	abp_aux_user
Entidades	GET	abp_entity_get
	POST	abp_entity_comment
	POST	abp_entity_enroll
	POST	abp_entity_like
	POST	abp_entity_unlike
	GET	abp_entity_likes_count
	GET	abp_entity_likes_users

E-Learning	GET	abp_learning_get
	POST	abp_learning_post_answers
Listas	GET	abp_list_courses
	GET	abp_list_entity
	GET	abp_list_users
	GET	abp_list_activity
Mensajes	GET	abp_message_count
	GET	abp_message_inbox
	GET	abp_message_read
	GET	abp_message_sent
	POST	abp_message_send

Se adjunta en el Anexo C la documentación completa asociada a la API. En él se encuentran todas las funciones, su descripción, parámetros de entrada, formatos de respuesta y ejemplos en JSON de respuestas de la API.

5.2. AFRICA BUILD App

Tal y como se ha comentado en apartados anteriores, la aplicación está orientada para dispositivos Android. Se ha decidido el uso de dicho sistema a causa de su extensión por todo el mundo, especialmente dentro del continente Africano, siendo también la decisión más económica y accesible.

5.2.1. Diseño

Análogo al diseño anterior, aquí se expone el realizado para la App. El diseño se define mediante un diagrama de casos de uso. Se argumentan después los casos de uso uno a uno siguiendo una lista de eventos.

5.2.1.1. Actores

Se definen los siguientes actores que interactúan en la solución:

- **Usuario**: Consiste en toda persona registrada en el ABP. El objetivo de los usuarios es aprovechar las actividades y recursos que ofrece el portal a través de los cursos y participar activamente en la comunidad de los mismos.
- **API**: Capa de servicios web con funcionalidad expuesta. Con la API se puede mandar al exterior toda la información que se solicite si cumple los requisitos de acceso y autenticación exigidos por el ABP.
- **App**: Sistema con el que interacciona el usuario. El objetivo de la App es mediar entre el usuario y el ABP en el intercambio de información. Para ello, la App se comunica con la API del ABP a través de consultas para solicitar y/o enviar la información para luego mostrarla al usuario.
- **Sistemas externos**: Corresponde a los sistemas auxiliares que ayudan a visualizar los recursos de aprendizaje como videos, webcasts y ficheros los cuales son accesibles a través de URLs.

5.2.1.2. Resumen del ámbito

La App debe disponer de la funcionalidad relacionada al módulo de "e-learning" del ABP. Para ello, el usuario debe poder acceder a los cursos, interactuar su contenido y poder participar en la comunidad de cada curso matriculado. Es importante destacar de nuevo que la App debe respetar las restricciones impuestas a los usuarios registrados en el ABP. Se muestra a continuación una lista más desglosada del ámbito de la solución:

- Autenticación de usuario: Obligatorio para permitir el acceso al portal.
- Cursos disponibles.
- Contenido global del curso
- Matriculación y baja de un curso.
- Comunidad del curso (comentarios, likes y actividad)
- Recursos de aprendizaje
    - Visualización de contenido (Videos, ficheros y webcasts).
    - Actividades: Quizzes

El ámbito de la función es preliminar y lo mínimo exigido para considerar terminada la solución. Sin embargo, la solución puede ser ampliada en futuras revisiones, aumentando así su ámbito y funcionalidad.

5.2.1.3. Diagrama de Casos de Uso

Se presenta a continuación el diagrama de casos de uso de la solución. Para una mejor comprensión del diagrama, este se divide en función de la interacción entre la App y otro de los actores. Primero se muestra el diagrama de casos de uso asociado a la interacción ABP-App (Fig. 29). El segundo diagrama corresponde a la interacción App-Usuario (Fig. 30).

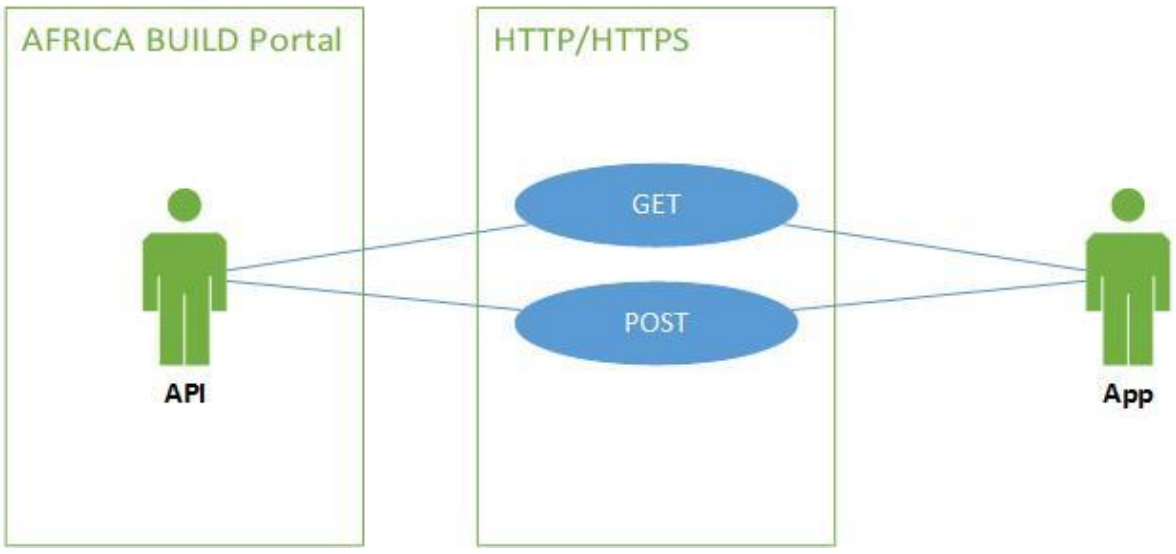


Fig. 29 Diagrama de Casos de Uso: API-App

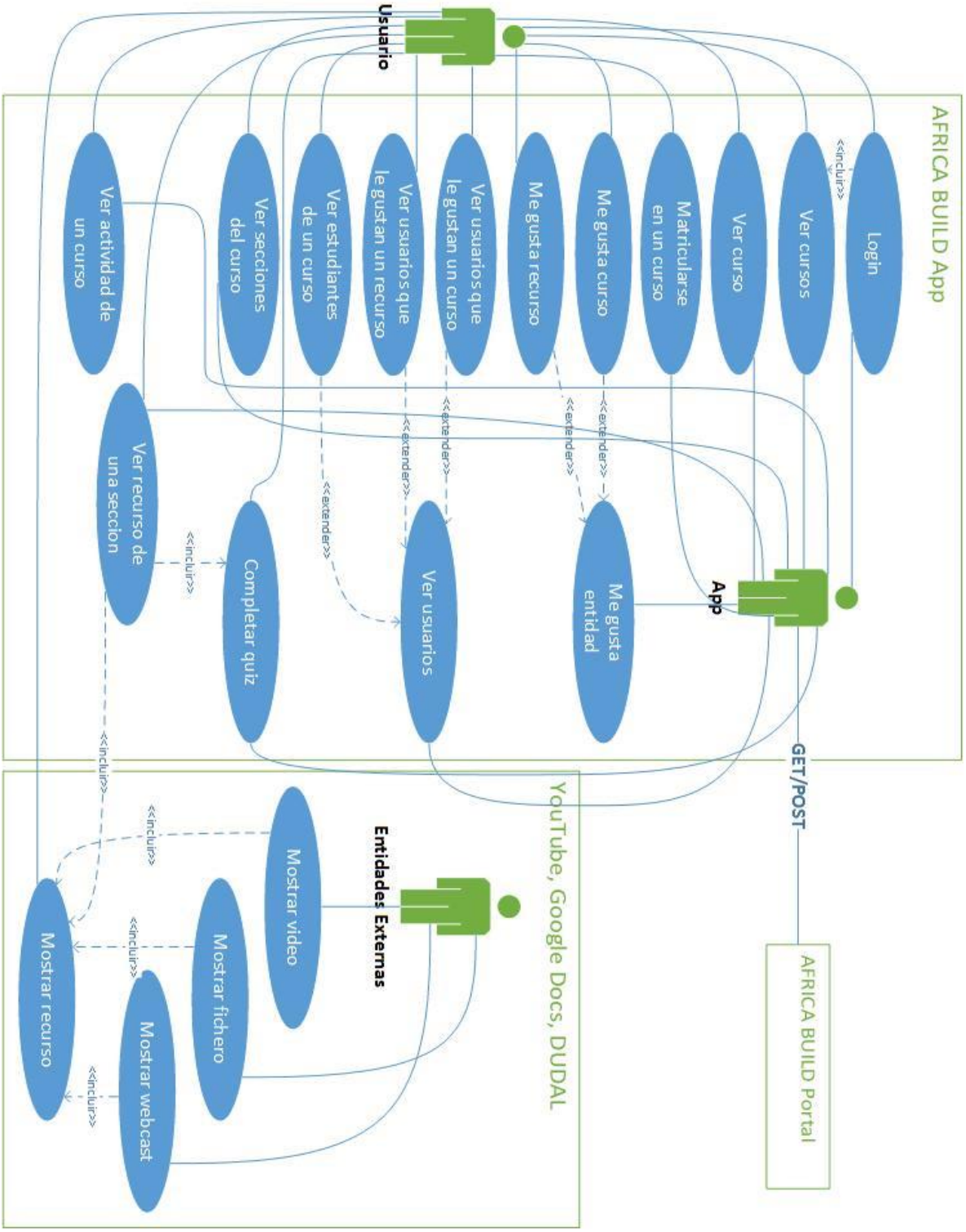


Fig. 30 Diagrama de casos de uso: Usuario-App-Entidades Externas

5.2.1.4. Casos de Uso

Se describen a continuación todos los casos de uso definidos en el diagrama. Los casos de uso son definidos en alto nivel y marcan la pauta del comportamiento del sistema frente a las peticiones del usuario.

I. Caso de uso **"GET"**

- Propósito: Petición GET del protocolo HTTP. Sirve para realizar una consulta al ABP a través de la API. Su objetivo consiste en proporcionar toda la información de las que se sustentan la mayoría de las peticiones del usuario en la App.
- Actores: App y API
- Precondición: Protocolo HTTP habilitado y conexión habilitada al ABP.
- Postcondición: Registro de la petición en el servidor del ABP.
- Eventos:
  1. La App, a petición del usuario, solicita una consulta al ABP.
  2. La App construye una URL con destino el sitio del portal. Dicha URL contiene el path del ABP y los parámetros necesarios para la consulta.
  3. La App envía la petición. La API recoge la consulta y reconoce el método y realiza la función expuesta si existe.
  4. La API devuelve en un formato serializado (XML o JSON) la respuesta.

II. Caso de uso **"POST"**

- Propósito: Petición POST del protocolo HTTP. Sirve para realizar una consulta al ABP de modificación o añadido de datos a través de la API. Su objetivo consiste en mandar toda la información necesaria para las peticiones del usuario que la soliciten.
- Actores: App y API
- Precondición: Protocolo HTTP habilitado y conexión habilitada al ABP.
- Postcondición: Registro de la petición en el servidor del ABP.
- Eventos:
  1. La App, a petición del usuario, solicita un envió de información al ABP.
  2. La App construye una URL con destino el sitio del portal. Dicha URL contiene el path del ABP y los parámetros necesarios para la consulta.
  3. Se construye si procede, un objeto serializable con los datos que se quieran adjuntar.
  4. La App envía la petición. La API recoge la consulta y reconoce el método, el objeto serializado y realiza la función expuesta si existe. Si la función no existe o hay un error de parámetros se devuelve una excepción para notificar a la App.
  5. La API devuelve en un formato serializado (XML o JSON) la respuesta. En caso de no obtener datos se le notifica a la App.

III. Caso de uso **"Login"**

- Propósito: Autenticación del Usuario en el ABP. Obligatorio para el acceso del contenido.
- Actores: Usuario y App.
- Precondición: Usuario registrado en el ABP.
- Postcondición: Usuario logueado en el ABP.

- Eventos:
  1. El usuario solicita entrar en el sistema mediante su nombre de usuario y contraseña.
  2. La App envía una petición de Login al ABP con los datos introducidos como parámetros.
  3. Si el Login ha sido correcto, la App se lo notifica al usuario. En caso contrario se deniega el acceso y se notifica al usuario.
  4. Salto al caso de uso "Ver cursos".

IV. Caso de uso **"Ver cursos"**

- Propósito: Proporciona una lista observable de los cursos disponibles dentro del ABP. Los cursos, al igual que en el portal, deben estar divididos en tres categorías: Todos, cursos propios y cursos matriculados.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP.
- Postcondición: -
- Eventos:
  1. Una vez logueado, la App solicita una petición a la API con la lista de todos los cursos, otra petición para la lista de cursos propios y una última lista de cursos matriculados.
  2. La App al recibir el resultado deserializa la respuesta y lista los cursos obtenidos.

V. Caso de uso **"Ver curso"**

- Propósito: Proporciona un resumen del curso, así como numero de likes, valor del like del usuario asociado al curso y acceso a su contenido y actividad si está matriculado.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP.
- Postcondición: -
- Eventos:
  1. El usuario escoge un curso de la lista
  2. La App solicita una petición de información del curso.
  3. La App recoge la respuesta y muestra al usuario la información del curso. Si el usuario está matriculado, se le habilitan los casos de uso de "ver secciones del curso" y "ver actividad del curso".

VI. Caso de uso **"Matricularse en un curso"**

- Propósito: El objetivo del caso de uso es habilitar la funcionalidad de matriculación del portal para el usuario. Si el usuario está matriculado en el curso puede acceder a la actividad y los recursos de aprendizaje del mismo
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP.
- Postcondición: Relación de matrícula entre el usuario y la entidad curso creada/borrada. Actualización de la actividad asociada al curso si el usuario se matricula.
- Eventos:
  1. El usuario, dentro de la vista de curso, solicita matricularse/darse de baja del mismo.
  2. La App realiza una petición de matrícula/baja al ABP.

3. La App recoge la respuesta y muestra al usuario la información del curso. Si el usuario está matriculado, se le habilitan los casos de uso de "ver secciones del curso" y "ver actividad del curso".

VII. Caso de uso **"Me gusta entidad"**

- Propósito: Solicita una petición de "like" para una entidad de tipo curso o recurso.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP.
- Postcondición: Añadido o borrado de la relación "like" entre la entidad y el usuario.
- Eventos:
  1. El usuario solicita una petición de "me gusta" o "like" a un curso o recurso de aprendizaje.
  2. La App manda la petición al ABP.
  3. Tras la respuesta, modifica visualmente el valor del "like" y actualiza el valor del número de "likes" totales de la entidad.

VIII. Caso de uso **"Ver usuarios"**

- Propósito: El objetivo del caso de uso consiste en mostrar una lista de usuarios con una relación con alguna entidad. Dicha relación puede ser de matriculados para ver los estudiantes de un curso o de "me gusta" ver los usuarios que les gusta un curso o recurso.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP.
- Postcondición: -
- Eventos:
  1. El usuario solicita una petición de lista de usuarios matriculados a un curso o usuarios que le gustan un curso o un recurso.
  2. El App construye la petición y la manda al ABP.
  3. Tras recibir la respuesta, la App muestra al usuario la lista visible con los usuarios obtenidos.

IX. Caso de uso **"Ver secciones del curso"**

- Propósito: El objetivo del caso de uso es presentar la lista completa de secciones del curso junto con la lista de recursos asociados a cada una de las secciones.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso.
- Postcondición: -
- Eventos:
  1. El usuario solicita acceder a las secciones de un curso.
  2. La App solicita una petición de secciones del curso al ABP.
  3. Tras la respuesta, la App indexa y muestra la lista de secciones.
  4. Si existen secciones, el usuario puede seleccionar una de ellas. Por defecto se selecciona siempre la primera sección.
  5. La App solicita una petición de lista de recursos de la sección al ABP.

6. La App indexa y muestra la respuesta.
7. Si el usuario quiere cambiar de sección, se repite el proceso desde el evento 5.

X. Caso de uso **"Ver actividad de un curso"**

- Propósito: El caso de uso muestra la actividad más reciente asociada a un curso. En dicha actividad se pueden ver sucesos como nuevos alumnos matriculados, comentarios y "likes" enviados, nuevo contenido y avisos entre otros.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso.
- Postcondición: -
- Eventos:
  1. El usuario solicita ver la actividad reciente dentro de un curso.
  2. La App realiza una petición de actividad al ABP.
  3. Tras la respuesta, indexa y muestra una lista de actividad ordenada de más reciente a menos reciente asociada al curso.

XI. Caso de uso **"Ver recurso de una sección"**

- Propósito: Análogo al caso de uso "ver curso", muestra un resumen del recurso de aprendizaje. Dicho resumen contiene su descripción, tipo, "likes" y el acceso a la visualización del recurso como tal.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso.
- Postcondición: -
- Eventos:
  1. El usuario selecciona un recurso de la lista de recursos de una sección seleccionada.
  2. El sistema solicita una petición al ABP para recibir la información del recurso.
  3. El sistema muestra el contenido de la respuesta al usuario.
  4. Si el recurso es de tipo video, webcast o fichero. Enlaza el acceso al recurso con la redirección a la URL del mismo.

XII. Caso de uso **"Completar quiz"**

- Propósito: El caso de uso consiste en mostrar al usuario el contenido del recurso de aprendizaje de tipo quiz. En él se presentan una serie de preguntas de diferente tipo y sus controles donde responder. Una vez finalizado el quiz o el tiempo máximo para completarlo, las respuestas se envían al ABP y este devuelve la corrección del mismo.
- Actores: Usuario y App.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso.
- Postcondición: - Actualización de la evaluación del usuario dentro del curso.
- Eventos:
  1. El usuario dentro de la vista resumen del recurso selecciona realizar la actividad.
  2. La App solicita una petición de datos del quiz al ABP.
  3. La App tras recibir la respuesta formatea el quiz y lo muestra al usuario.
  4. El usuario responde a las preguntas dentro del tiempo estimado en el quiz (si procede).

5. Al terminar el quiz o el tiempo máximo se deshabilita el responder más preguntas.
6. Las soluciones propuestas se instancian en un objeto serializable y se mandan al ABP junto con una petición de corrección.
7. La App recoge la corrección del quiz y muestra los resultados al usuario.

XIII. Caso de uso **"Mostrar recurso"**

- Propósito: El caso de uso consiste en mostrar al usuario el contenido del recurso de aprendizaje de tipo video, webcast o fichero. Según el tipo de recurso a mostrar se realiza una redirección a la entidad externa competente (YouTube para videos, Google Docs para ficheros y DUDAL para los webcasts). Ellos cargan vía web su contenido (a través del navegador web del dispositivo móvil), abstrayendo la App del tratamiento del recurso.
- Actores: Usuario y Entidad Externa.
- Precondición: Usuario logueado en el ABP. Usuario matriculado en el curso.
- Postcondición: -
- Eventos:
  1. El usuario dentro de la vista resumen del recurso selecciona realizar la actividad.
  2. La App redirige la actividad al navegador web del dispositivo móvil con la URL a utilizar.
  3. El navegador ubica la dirección y enlaza a la entidad externa que se encarga de visualizar el contenido del recurso.
     - YouTube para videos
     - DUDAL para webcasts
     - Google Docs Viewer para ficheros.

5.2.2. Implementación

En el apartado de implementación se argumentan todas las decisiones y procesos realizados para la producción de la solución. No se presenta en este apartado el código de la APP por motivos de propiedad intelectual. Sin embargo, si se muestran breves fragmentos del módulo de serialización y peticiones a la API para una mejor comprensión de lo expuesto.

5.2.2.1. Versionado

Antes de implementar, se debe determinar la API de Android mínima y la API usada para el desarrollo de la App (Ver apartado de Tecnologías Empleadas – SO Android). La decisión implica tanto el diseño de las vistas y su usabilidad como la cantidad de dispositivos. En los requisitos se exige la compatibilidad desde la API 8 (Android 2.2). Esta decisión es tomada ya que se considera una API mínima que abarca el 99% de los dispositivos Android actualmente en uso (Ver Fig. 24). Por otro lado, la API 8 en comparación con las actuales (API 16 hasta la 19) no supone problemas de usabilidad ni de diseño que comprometan el correcto uso de la App.

Dicho esto, se decide implementar la App con la API 19 de Android, la más actual hasta el momento; correspondiente a la versión KIT-KAT 4.4.2. La API mínima compatible corresponde a la API 8 o FROYO 2.2.

5.2.2.2. Estructura y definición de Vistas

Se muestra a continuación el esquema general de vistas con el que se puede recorrer toda la App. Más tarde se definen cada una de las vistas y sus componentes.

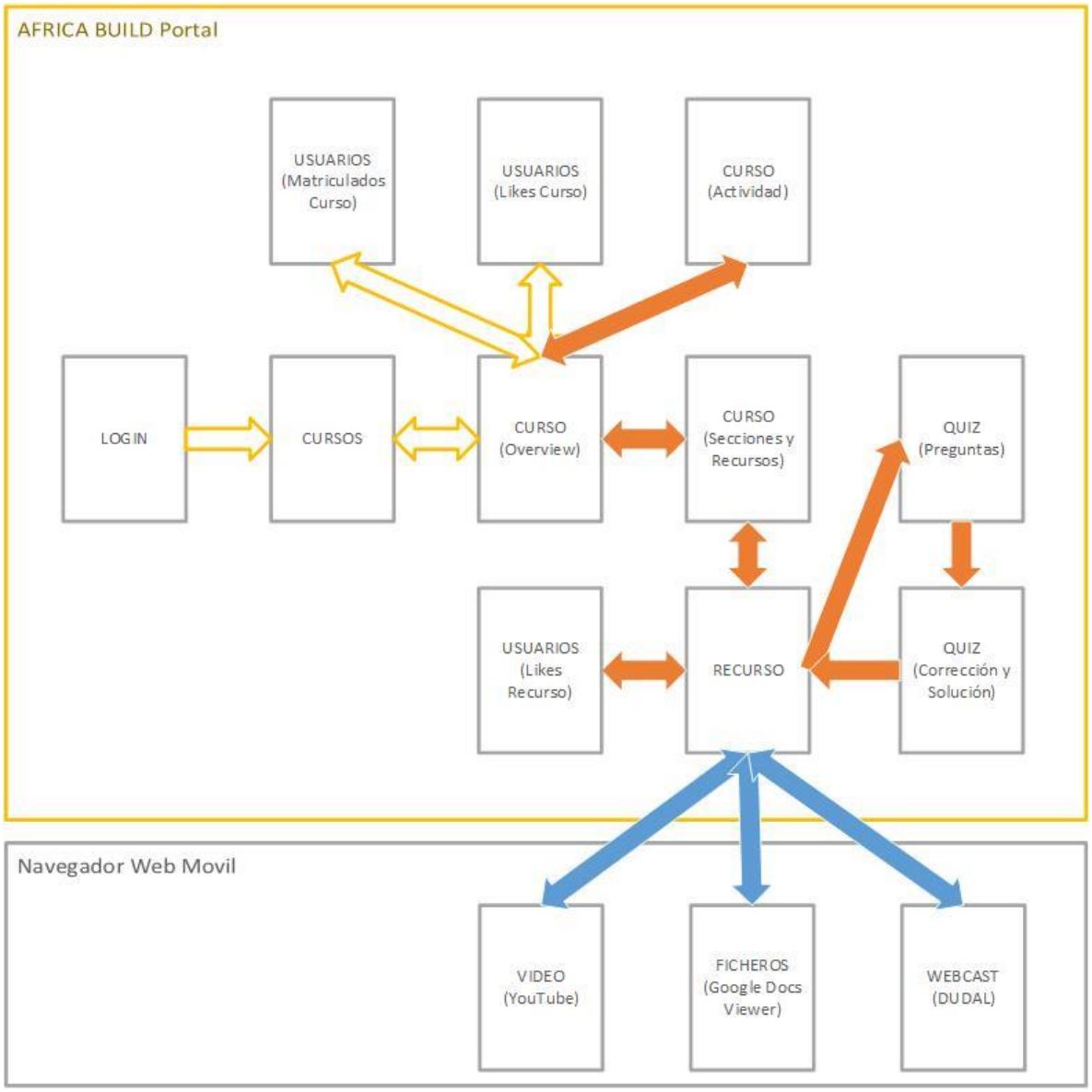


Fig. Esquema de comunicación entre vistas en la App

Con respecto a la figura, se describen dos sistemas: la App en amarillo y el navegador web del dispositivo móvil en azul. Las transiciones en blanco solo necesitan la condición de haber completado el login para poder visualizarlas. Las transiciones en naranja solo pueden realizarse si el usuario está matriculado en el curso (Desde la vista resumen o "overview" del curso). Finalmente,

las transiciones en azul corresponden al paso entre sistemas; en este caso, entre la App y el navegador web.

- **Login**: Consiste en la vista de bienvenida del portal. En él se introducen los parámetros para la autenticación del usuario.

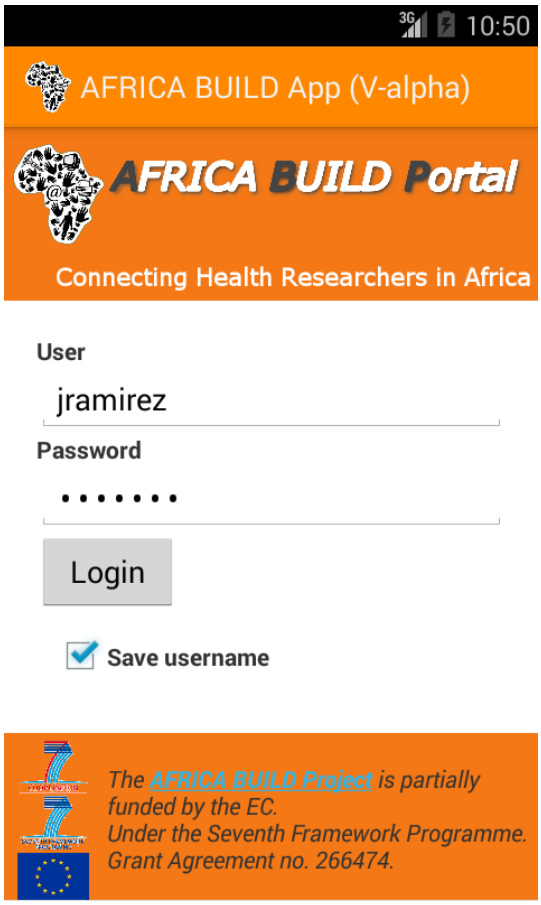


Fig. 31 Vista de Login

- **Cursos**: La vista general de cursos. Los cursos están separados en tres listas: Todos los cursos, cursos propios y cursos matriculados. Se enseña información breve de cada curso: Foto, nombre, descripción y fecha de creación.

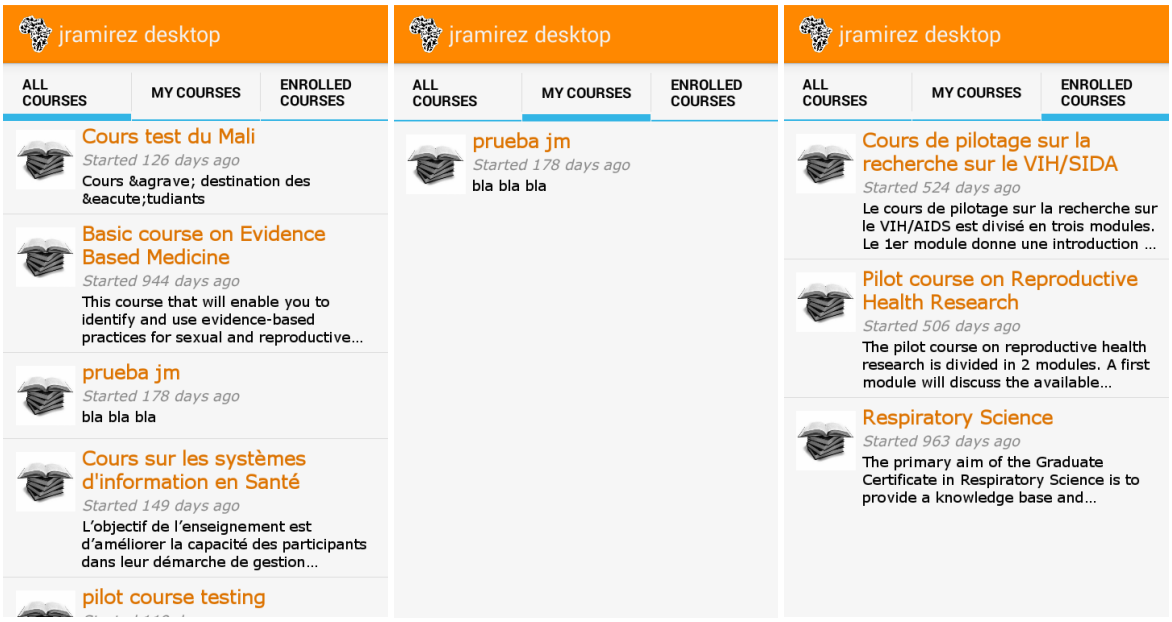


Fig. 32 Vista de Cursos: Todos, Mi cursos y Cursos matriculados.

- **Curso (Overview)**: Vista de curso. En él se contempla la siguiente información: Titulo, fecha de creación, descripción, el número de likes y el estado del "like" del usuario y si está matriculado. La vista proporciona acceso a las vistas de sección, actividad, estudiantes y usuarios que les gusta el curso.

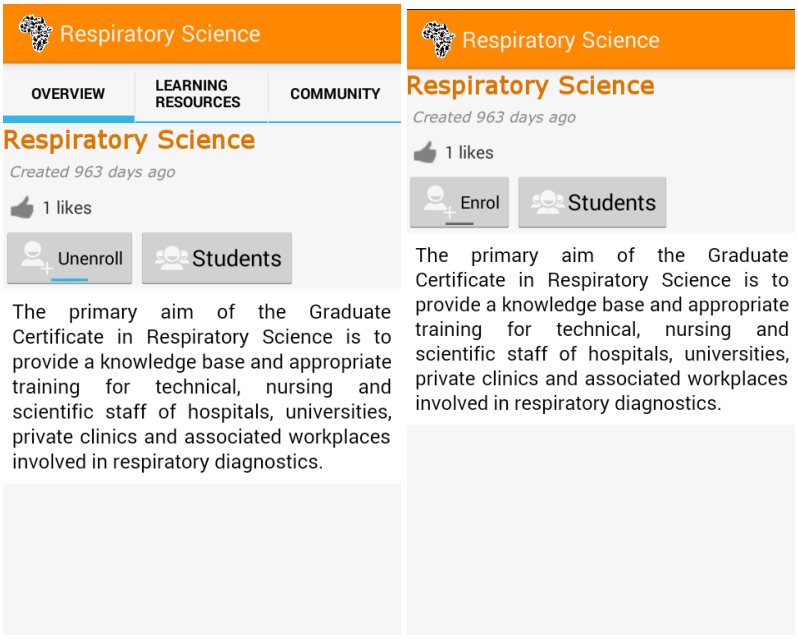


Fig. 33 Vista de Curso (Matriculado y sin matricular).

- **Curso (Secciones y recursos)**: En él se visualizan la lista de recursos por sección. Se dispone de un menú desplegable donde se puede seleccionar la sección. Una vez seleccionada la sección, la lista de recursos se actualiza automáticamente.

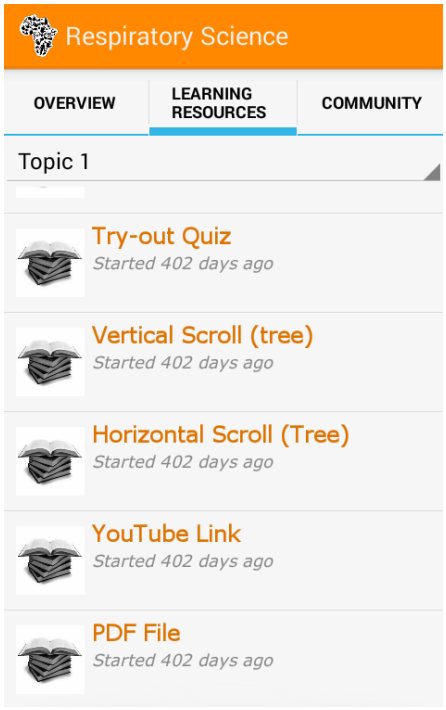


Fig. 34 Vista de Curso: Secciones y recursos.

- **Curso (Activity)**: En esta vista se muestra la lista de actividad más reciente. Cada elemento de la lista contiene un comentario de la actividad realizada dentro del curso con el usuario, la acción, el objetivo de la acción y la fecha. Se plantea en una próxima ampliación dar acceso desde esta vista al resto del contenido asociado a la comunidad del curso: Blogs, Páginas, Bookmarks, Discusiones y Ficheros.

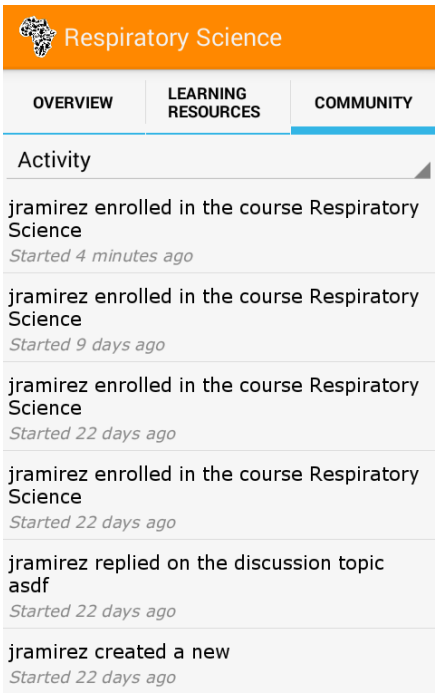


Fig. 35 Vista de Actividad

- **Usuarios (Likes curso/recurso)**: Análogo para cursos y recursos, presenta una lista de usuarios que le han dado a "like" a la entidad y el número de usuarios totales. Se devuelve por cada elemento la foto y el nombre de usuario.

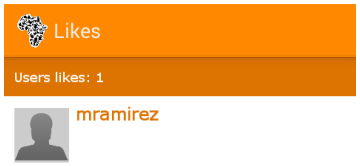


Fig. 36 Vista de Usuarios (Likes)

- **Usuarios (Matriculados curso)**: Proporciona la lista completa de usuarios matriculados dentro del curso y además del conteo de usuarios. Cada elemento se muestra una foto, su nombre de usuario y nombre completo.

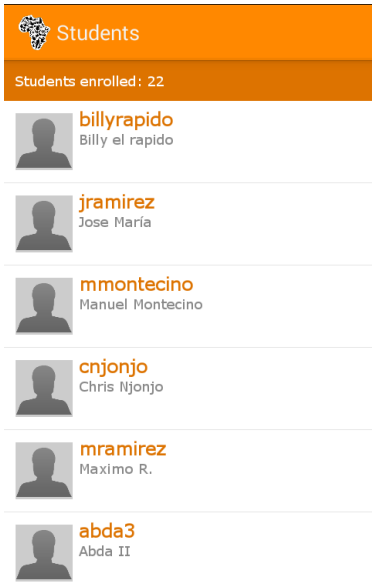


Fig. 37 Vista de Usuarios matriculados

- **Recurso**: Vista que contiene la información general del recurso: Titulo, fecha de creación, numero de “likes” y valor “like” del usuario. Desde esta vista se accede al recurso, ya sea un quiz o una redirección al navegador web.

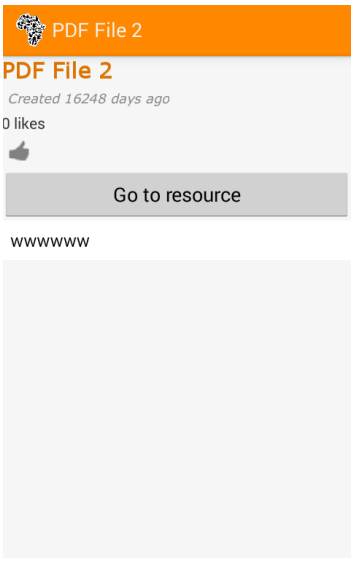


Fig. 38 Vista de Recurso

- **Video, Fichero y Webcast**: La vista del navegador es independiente de la App; es decir, el navegador web móvil es responsable de la visualización y adaptación del contenido.

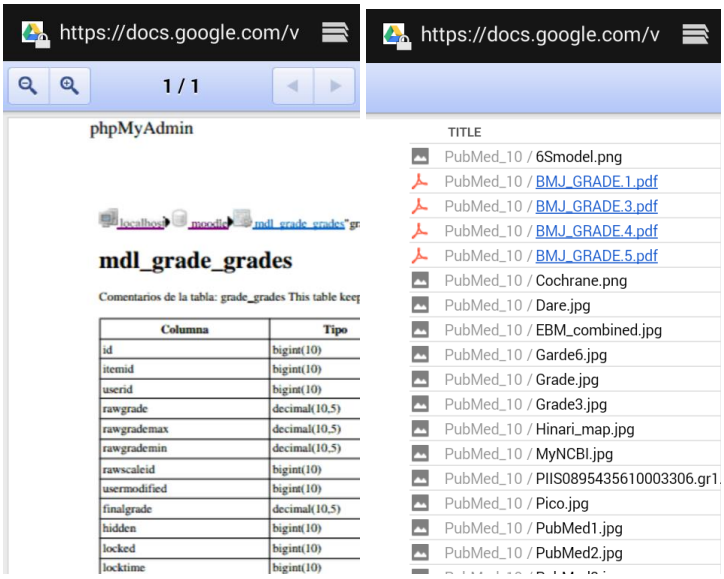


Fig. 39 Vista de Recurso desde el Navegador Web: Fichero .pdf y .zip

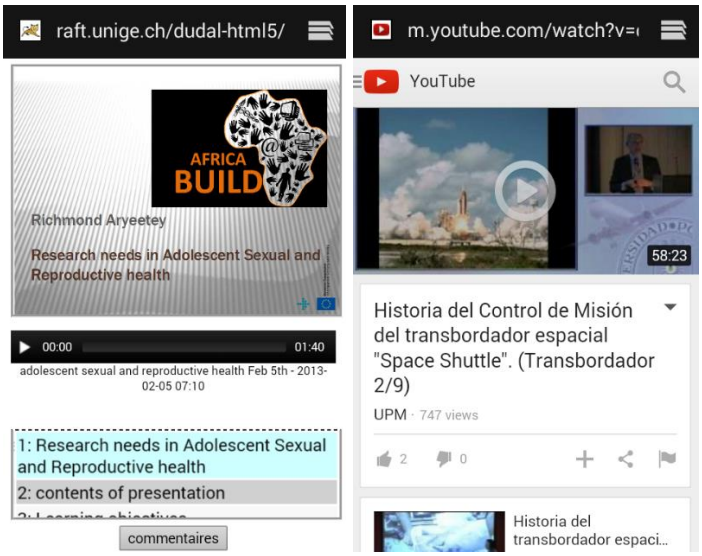


Fig. 40 Vista de Recurso desde el Navegador Web: Webcast y Video.

- **Quiz (Preguntas)**: En esta vista se responden a las preguntas del quiz. En la parte superior de la vista se muestra el tiempo restante para completar el quiz (“No time” si no procede). En la parte inferior está el botón que finaliza el quiz manualmente y envía las repuestas. Tras

él envió, el usuario queda a la espera de recibir la solución. Una vez recibida se le redirige a la vista de corrección y soluciones.

Existen diversos tipos de preguntas, las cuales tienen una visualización especifica: Un control de texto para las preguntas que se contestan con teclado (más ancho para las preguntas de tipo desarrollo), checkbox para las preguntas de selección múltiple, botones radio para las preguntas de selección única y menús desplegables con las opciones disponibles para las preguntas de "unir correctamente".

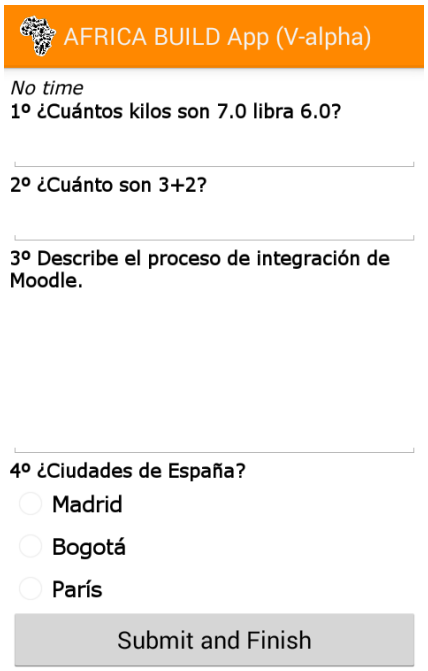


Fig. 41 Vista de Quiz: Preguntas.

- **Quiz (Corrección y Solución) [EN PROCESO]**: En esta vista se muestra la realimentación del quiz completado. Se visualiza la puntuación y soluciones a cada pregunta.

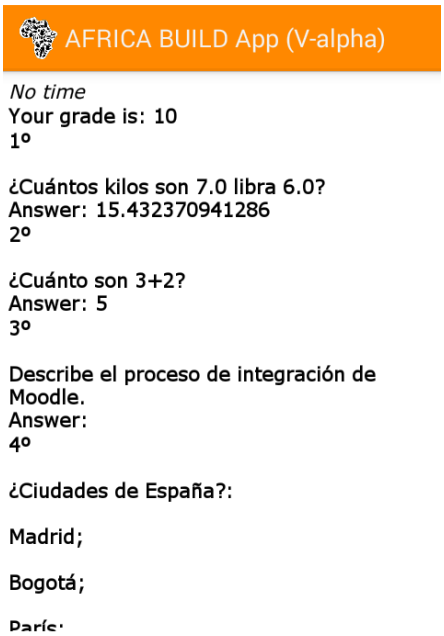


Fig. 42 Vista de Quiz: Corrección y Respuestas

5.2.2.3. Transición entre Vistas

En este apartado se describe la transición entre vistas dentro de la con la API. Todas las peticiones se realizan de manera asíncrona a la carga de contenido de la App. El objetivo consiste en favorecer la carga y reducir el tiempo de espera mediante la ejecución simultanea de dos hilos: uno para la visualización y otro para el tratamiento de la petición y recepción de la respuesta. Por cuestiones de usabilidad e integración de los datos, se toma la decisión de bloquear el uso de la App hasta que todo su contenido esté disponible. Esto se debe a que se puede dar el caso de que el hilo de visualización termine antes que el de tratamiento de los datos. Esto provoca que el usuario maneje información incorrecta y pueda provocar una discrepancia entre datos.

Para solucionarlo, se espera al hilo de tratamiento de datos antes de habilitar la vista. Mientras que este hilo no termine (haya terminado o no el de visualización) se debe bloquear la interacción con la App. Se muestra así una ventana de progreso (Fig. 43) que desaparece en cuanto los dos hilos hayan terminado.

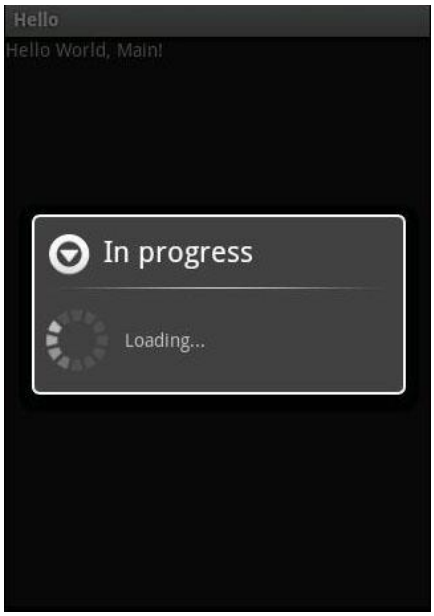


Fig. 43 Ventana de espera.

5.2.2.4. Serialización y parsing

En este apartado se describe el módulo de tratamiento de las peticiones a la API. Tal y como se ha definido en el apartado anterior, las llamadas a la API se realizan de manera asíncrona. Se debe definir para realizar una petición una clase que extienda "AsyncTask", clase propia de Android para la ejecución de hilos en segundo plano.

```
private class ElggAPICall extends AsyncTask<String, Integer, Boolean> {

        @Override
        protected Boolean doInBackground(String... params) { … }

        @Override
        protected void onPostExecute(Boolean result) { … }

}
```

En Android, la clase que extiende "AsyncTask" debe definir mínimo las funciones "doInBackground()" y "onPostExecute()". La primera realiza la lógica de la petición en segundo plano y la segunda se ejecuta cuando "doInBackground()" ha finalizado y señaliza el fin el hilo.

Para inicializar la tarea de petición, se debe crear un objeto de la clase definida y llamar a la función heredada "execute()" con los parámetros que se deseen. Los parámetros son recogidos como el atributo "params" en "doInBackground()". En el momento de la llamada se inicializa el hilo y debe hacerse desde el hilo principal ya sea capturando la pulsación de un botón o al inicio de la carga de la visualización.

```
Button submit = (Button)findViewById(R.id.resource_submit);
submit.setOnClickListener(new OnClickListener() {

	@Override
	public void onClick(View v) {
		ElggAPICall answer = new ElggAPICall();
		answer.execute("data1","data2","data3");
	}
});
```

La serialización de la información a mandar, seguido de la obtención de los datos de la API y su parsing se realizan dentro de la función "doInBackground()". En esta función también se procede de enviar los datos a los controles de la vista a la que pertenece la clase.

A continuación se muestra un código de ejemplo dentro del método "doInBackground()". En dicho ejemplo se realiza una petición GET a la función "abp_list_users" de la API para obtener una lista de usuarios asociados a una entidad (Ver Anexo C). Los parámetros a incorporar a la URL son: el identificador de la entidad contenedor, el tamaño de la lista, el índice de búsqueda y el token de identificación de usuario. Android proporciona las librerías en Java con las clases "HttpClient" para configurar el cliente y las clases HttpGet y HttpPost para instanciar las peticiones GET y POST en HTTP [54].

```
boolean ok = true;

HttpClient httpClient = new DefaultHttpClient();
HttpGet getCourses = new HttpGet(

	http://egasmoniz.dia.fi.upm.es/elgg/services/api/rest/json/?method=abp_list_users&id_entity= + params[0] + "&limit=" + params[1] + "&offset=" + params[2] + "&auth_token=" + params[3]);
```

El objeto de la clase "HttpGet" o "HttpPost" debe inicializarse con la URL de la función a la que se quiere llamar de la API. En nuestro caso, los parámetros que sirven de filtro para la petición se cogen del atributo de entrada "params" de la función. En la URL hemos definido el recurso "json" para recibir en formato JSON la respuesta a la petición. Este modo de establecer el formato de respuesta es definido por Elgg (Ver Anexo A).

```
getCourses.setHeader("content-type", "application/json");
ActivityStudentsCurso.students = new ArrayList<Usuario>();

try {
	HttpResponse response = httpClient.execute(getCourses);
	String respStr = EntityUtils.toString(response.getEntity());

	JSONObject resp = new JSONObject(respStr);
	JSONArray result = resp.getJSONArray("result");

	int size = result.length();
```

```
        for (int i = 0; i < size; i++) {
                JSONObject obj = result.getJSONObject(i);
                Usuario myUser = new Usuario(obj.getString("username"),
                obj.getString("name"), obj.getString("email"),
                obj.getString("avatar_url"));

                ActivityStudentsCurso.students.add(myUser);
        }
```

La función “execute()” del objeto de la clase “HttpGet/HttpPost” lanza la petición y el hilo se queda esperando a la respuesta, la cual se instancia en un objeto de la clase String. La respuesta recibida debe deserializarse y parsearse para poder extraer los datos. La clase “JSONObject” de Java se encarga de instanciar la información. El parsing del objeto “JSONObject” respuesta debe hacerse de manera manual extrayendo la información a través de las funciones get() de “JSONObject” y pasando como parámetro el nombre de la etiqueta que se quiere extraer.

En nuestro caso, instanciamos en un objeto de la clase “JSONArray” el contenido con la etiqueta “result”, donde se ubica el contenido de respuesta de cualquier petición realizada a la API (Ver Anexo C). Al ser un array de contenido, se saca uno a uno cada elemento de la estructura identificando los elementos a partir de su etiqueta y se instancia en la estructura de datos que deseemos para poder lanzarla a la vista.

```
} catch (ClientProtocolException e) {
        this.error = "Client Protocol Error";
        Log.e(this.error, e.toString());
        ok = false;
} catch (IOException e) {
        this.error = "IO Error";
        Log.e("IO Error", e.toString());
        ok = false;
} catch (JSONException e) {
        this.error = "JSON Error";
        Log.e(this.error, e.toString());
        ok = false;
}
```

Se debe controlar los errores que puedan surgir durante la petición. En nuestro caso, controlamos los errores asociados a un fallo en el uso del protocolo HTTP, fallo de Entrada/Salida y fallo de uso de las clases que heredan la librería JSON de Java. En caso de error, se debe notificar al hilo principal lo sucedido. El hilo principal se encargará entonces de avisar al usuario y parar la ejecución del programa para evitar problemas y mantener la estabilidad del dispositivo.

```
runOnUiThread(new Runnable() {

        @Override
        public void run() {
                TextView numStudents = (TextView)findViewById(R.id.LblNumStudents);
```

```
                numStudents.setText("Students enrolled: " + students.size());
                …
        }
});

return ok;
```

Finalmente, si la estructura de datos auxiliar se ha llenado correctamente, la función "runOnUiThread()" se comunica con el hilo principal y ejecuta lo contenido dentro de la función "run()" de la clase "Runnable". En nuestro caso, la función se encarga de identificar e instanciar el controlador al que queremos "inflarle" de información a mostrar en la vista.

6. PRUEBAS Y EVALUACIÓN

En este capítulo se desglosa el plan de pruebas realizado. Para ello el plan se divide en dos partes, cada una de ellas está asociada a una de las soluciones y tiene un conjunto de pruebas propio. En el caso de los servicios Web, las pruebas se centran más en la funcionalidad mientras que en la App se prioriza la usabilidad y la integración en diferentes dispositivos.

6.1. Servicios Web

6.1.1. Pruebas unitarias

Para poder probar la funcionalidad de la API, se han desarrollado un conjunto de pruebas unitarias para cada función expuesta. Las pruebas determinan el correcto funcionamiento de la solución elemento a elemento. Las pruebas definidas para cada función comprenden los siguientes casos:

- Dos pruebas que devuelvan respuesta correcta.
- Una prueba con parámetros erróneos.
- Una prueba sin resultado.

Se adjunta a continuación la tabla resumen con la muestra de resultados por cada función testada. La resolución de los errores ha sido bastante sencilla ya que la modularidad de los bloques junto a la ausencia de acoplamiento entre las funciones expuestas acota en gran medida la ubicación del error. La única función que sigue sufriendo errores aún está en proceso de implementación y la mas complicada de toda la API (abp_learning_post_answers).

Bloque	Función	Método	Nº pruebas	Pruebas correctas	Pruebas fallidas	% Error
Funciones Auxiliares	GET	abp_aux_hello	4	4	0	0
	GET	abp_aux_site	4	4	0	0
	GET	system.api.list	4	4	0	0
	POST	auth.gettoken	4	4	0	0
	GET	abp_aux_user	4	4	0	0
Entidades	GET	abp_entity_get	4	4	0	0
	POST	abp_entity_comment	4	4	0	0
	POST	abp_entity_enroll	4	4	0	0
	POST	abp_entity_like	4	4	0	0
	POST	abp_entity_unlike	4	4	0	0
	GET	abp_entity_likes_count	4	4	0	0
	GET	abp_entity_likes_users	4	4	0	0
E-Learning	GET	abp_learning_get	4	4	0	0
	POST	abp_learning_post_answers	4	2	2	50
Listas	GET	abp_list_courses	4	4	0	0
	GET	abp_list_entity	4	4	0	0
	GET	abp_list_users	4	4	0	0
	GET	abp_list_activity	4	4	0	0

Mensajes	GET	abp_message_count	4	4	0	0
	GET	abp_message_inbox	4	4	0	0
	GET	abp_message_read	4	4	0	0
	GET	abp_message_sent	4	4	0	0
	POST	abp_message_send	4	4	0	0

6.1.2. Pruebas de Integración

Se han definido una serie de pruebas de integración que permiten analizar la cohesión entre los bloques de la API. Para ello se ha diseñado un proceso de pruebas iterativo e incremental. Dicho proceso consiste en añadir un bloque a producción una vez implementado y si este supera las pruebas unitarias y de integración. Entonces, el siguiente bloque implementado deberá pasar sus pruebas unitarias y después realizar las pruebas de integración con el resto del producto en producción.

Además, todas las pruebas realizadas deben ser cotejadas desde la versión web del portal para comprobar también su integración con la misma. Así podemos analizar la integridad y consistencia de los datos que viajan por la red.

Se adjunta a continuación la tabla de pruebas de integración realizadas en cada iteración. En la tabla se enfrentan las pruebas fallidas con las realizadas en cada iteración. En la última iteración podemos tenemos el problema de que no se están pasando correctamente las respuestas de algunos tipos de preguntas del quiz. La funcionalidad asociada a los quizzes aún está en desarrollo y no será puesta en producción hasta que pase todas las pruebas de sistema e integración.

<table>
<tr><th>Bloque</th><th>Iteración 1 (Fallidas/Realizadas)</th><th>Iteración 2 (F/R)</th><th>Iteración 3 (F/R)</th><th>Iteración 4 (F/R)</th><th>Iteración 5 (F/R)</th></tr>
<tr><td>Funciones Auxiliares</td><td>5/5</td><td rowspan="2">7/7</td><td rowspan="3">4/4</td><td rowspan="4">5/5</td><td rowspan="5">3/5</td></tr>
<tr><td>Entidades</td><td rowspan="4"></td></tr>
<tr><td>Mensajes</td><td rowspan="3"></td></tr>
<tr><td>Listas</td><td rowspan="2"></td></tr>
<tr><td>E-Learning</td><td></td></tr>
</table>

6.2. App

6.2.1. App – Compatibilidad con diferentes versiones de Android

Una de las principales pruebas que debe superar la App consiste en la compatibilidad de versiones. Se definió el requisito del versionado con el objetivo de maximizar la disponibilidad de la aplicación. Para ello, se han realizado pruebas de sistema en simuladores de smartphone desde la versión mínima a la más actual de Android. Para cada simulación se ha analizado sobre todo la visualización y la colocación del contenido. Valores como los tiempos de petición a la API y transiciones entre vistas no pueden medirse en una simulación ya que depende de las características del sistema que realiza la emulación y no del dispositivo simulado.

Se adjunta a continuación la tabla de pruebas realizada. En ella se especifican la versión probada, el tamaño de pantalla usado y su resolución, las vistas analizadas y si estas tienen problemas de visualización. Se han descartado las APIs que Android ya no da soporte o tienen una distribución de menos de un 0,1% (Ver Fig. 24). En esos casos, Android proporciona la actualización más reciente para el dispositivo.

Versión	Pantalla	Vista de Login	Vista de Listas de Cursos	Vistas de Curso	Vistas de quiz	Vistas de recursos URL
API 8 2.2	2,7” 240x320	OK	OK	OK	Fallo	OK
API 10 2.3.7	3,2” 320x480	OK	OK	OK	OK	OK
API 15 4.0.4	3,7” 480x800	OK	OK	OK	OK	OK
API 16 4.1.2	4” 480x800	OK	OK	OK	OK	OK
API 17 4.2.3	4,65” 720x1280	Fallo	Fallo	Fallo	-	OK
API 18 4.3	4” 480x800	OK	OK	OK	-	OK
API 19 4.4.2	4” 480x800	OK	OK	OK	-	OK

Las pruebas marcadas con un guion corresponden a pruebas aún no realizadas ya que la funcionalidad de devolver la respuesta asociada a un quiz está todavía en desarrollo. Las pruebas con fallo corresponden a controles descolocados o fuera de la pantalla, lo que supone un problema de usabilidad notable. En el caso de la API 8, el problema se encuentra a la hora de imprimir el contenido del quiz en una pantalla con poca resolución. En la API 17 los problemas surgen por todo lo contrario. El dispositivo corresponde a una Tableta y, aunque todo el contenido se muestra, se está desperdiciando mucho espacio útil para mostrar contenido. Está pendiente de desarrollo el diseño de la interfaz para dispositivos de alta resolución.

6.2.2. Pruebas de Sistema

Se ha diseñado un conjunto de pruebas de sistema para testear la funcionalidad y uso del App. Las pruebas de sistema consisten en la realización con éxito de cada uno de los casos de uso definidos en el diseño.

En la siguiente tabla se indican los casos de uso, las pruebas realizadas y fallidas junto con el porcentaje de error. Para los casos de uso GET y POST, se han realizado diferentes pruebas para cubrir todas las peticiones a la API.

Caso de uso	Pruebas Realizadas	Pruebas Correctas	Pruebas Fallidas	% Error
"GET"	32	32	0	0
"POST"	14	12	2	14%
"Login"	5	5	0	0
"Ver cursos"	6	6	0	0
"Ver curso"	5	5	0	0
"Matricularse en un curso"	6	6	0	0
"Me gusta entidad"	4	4	0	0
"Ver usuarios"	4	4	0	0
"Ver secciones del curso"	6	5	1	20%
"Ver actividad de un curso"	8	5	3	37%
"Ver recurso de una sección"	6	6	0	0
"Completar quiz"	4	2	2	50%
"Mostrar recurso"	6	6	0	0

Las pruebas de sistema a los casos de uso de "POST" y "Completar quiz" corresponden a la petición de enviar quiz con las soluciones del usuario. Tal y como se ha comentado anteriormente, este error ocurre a causa de un fallo en la serialización del envío de las respuestas de los tipos de preguntas "unir correctamente" y selección múltiple. Los errores asociados a los casos de uso "ver actividad" y "ver secciones" se debe a un bloqueo de la aplicación cuando la petición tarda demasiado en ser respondida para pruebas con conexiones lentas. Por lo demás podemos confirmar la efectividad del resto de la funcionalidad de la App, quedando pendiente unos pocos errores de cara a la próxima puesta en producción.

6.2.3. Test de Usabilidad

Queda pendiente de realizar un test de usabilidad cuando la App esté puesta en producción. Dicho test no puede realizarse hasta que toda la funcionalidad haya quedado completa y los errores de visualización para las diferentes resoluciones de smartphone queden arreglados. De todos modos, se adjunta a continuación el escenario planteado para el test, el cual determinará la usabilidad de la aplicación y si esta debe someterse a otra batería de cambios en su interfaz.

La solución propuesta se basa en un test sencillo de no más de cinco minutos, basado en una serie de instrucciones para que el usuario interactúe con la aplicación. Seguidamente se le realizan una serie de preguntas de respuesta corta donde podremos analizar la satisfacción del usuario. El test solo podrán hacerlo aquellos usuarios que estén registrados en el ABP. A continuación se enumeran las instrucciones a realizar:

- Hacer Login en el sistema, matricularse en un curso y ver los usuarios matriculados.
- Buscar otro curso y darle a "me gusta", matricularse y ver en la actividad las acciones realizadas.
- Desmatricularse del curso y ver su lista de cursos matriculados.
- Buscar un recurso de video en un curso, acceder a él y poder visualizarlo.
- Buscar un recurso de webcast en un curso, acceder a él y poder visualizarlo.
- Buscar un recurso de fichero en un curso, acceder a él y poder visualizarlo.
- Buscar un recurso de quiz en un curso, acceder a él completarlo y ver la respuesta.

Las preguntas a realizar tras la finalización del test son:

1. ¿Qué instrucciones no ha podido completar? ¿Por qué?
2. ¿Qué instrucción le ha parecido más fácil? ¿Y más difícil?
3. ¿Hay algún botón que no sabía lo que hacía? ¿Cuales?
4. Valora de 1 a 5 si la solución es intuitiva.
5. Valora de 1 a 5 si la solución le parece útil.
6. Valora de 1 a 5 la reacción del sistema.

Como indicadores adicionales, se hará un conteo de los siguientes conceptos durante la prueba:

- Nº de pulsaciones realizadas.
- Nº de pulsaciones erróneas.
- Tiempo.
- Nº de veces que el usuario solicita ayuda o se queda bloqueado.

7. CONCLUSIONES Y LÍNEAS FUTURAS

Para finalizar esta memoria, se argumentan las conclusiones obtenidas del desarrollo del trabajo, haciendo énfasis en las dificultades encontradas, logros obtenidos e impresiones; sin olvidar las tareas que aún quedan por hacer. Finalmente, se mencionan las posibles líneas futuras que puedan derivar de este trabajo ya sea para mejorarlo o ampliarlo.

7.1. Conclusiones

El AFRICA BUILD Portal ofrece una gran cantidad de contenido y recursos para los usuarios. Toda la información a mostrar es muy adecuada para dispositivos de mesa como ordenadores personales, portátiles, etc… Sin embargo, supone un problema cuando queremos visualizarlo en un dispositivo de mano o de pequeño tamaño; ya que resulta imposible poder mostrar la misma cantidad de información dentro de una pantalla que es del orden de más de cinco veces su tamaño.

Dicho esto, supone una tarea primordial y necesaria el seleccionar que información queremos priorizar y cual descartar. Durante el desarrollo de la versión alfa de la aplicación, se estuvo manejando esta tarea, analizando en diferentes tamaños de pantalla el contenido a mostrar. El objetivo consiste en conseguir un modelo de interfaz más heterogéneo posible con todas las resoluciones compatibles con la aplicación. Este objetivo se consiguió en parte ya que aún queda pendiente la distribución de la información para dispositivos como tabletas, las cuales tienen un mayor tamaño de pantalla y no pueden seguir el esquema de las pantallas de smartphone.

Uno de los problemas más graves que sufrió el desarrollo de la aplicación web consistió en la fase de desarrollo del módulo de serialización y parsing de las peticiones. En un principio se implementaron librerías de lectura de contenido en XML con procesos DOM y SAX. La solución parecía efectiva hasta que la aplicación fue probada en el continente africano. La deserialización del contenido, su parsing y adaptación a estructuras de datos válidas para su tratamiento era demasiado pesada y para conexiones por debajo del 3G suponía la inoperatividad del sistema. Por ejemplo, la respuesta a una petición de Login consiste en dos campos de menos de 50 caracteres; sin embargo, el tratamiento de la respuesta tardaba más de 15 segundos en completarse.

Para solucionar este problema, se tuvo que probar diferentes implementaciones de parsing y serialización. Fue de estas pruebas donde se encontró la instanciación en JSON. La solución fue muy efectiva; hasta tal punto de que en lugares con alta conectividad la petición, instanciación y visualización de contenido es casi inmediata.

Otro problema que surgió durante el diseño preliminar es la imitación de los estilos, dibujos y distribución de los elementos visibles del portal en la App. Uno de los objetivos de la aplicación era evitar que el usuario supiese diferenciar entre la App y el ABP. Esto llego a suponer un gran esfuerzo ya que la ordenación del contenido delataba con creces dicha diferencia. En la App los recursos se redirigen de manera completamente diferente al portal por limitaciones técnicas. Mientras que el ABP muestra el contenido de los recursos a través de una ventana que genera un plugin llamado “Whiteboard”, En la App se optó por hacer una redirección al navegador web del dispositivo para aquellos recursos que suponían un problema cargarlos dentro de las vistas.

Aun así, durante las sucesivas puestas en producción de la App, los usuarios que han estado probándola han comentado buenas impresiones y no dudan de su utilidad de cara al proyecto AFRICA BUILD. Muchos de los usuarios que han manejado la App han podido interactuar en mayor medida con el portal y les ha servido de ayuda en momentos puntuales donde no disponen de otro modo de acceso al mismo.

Por otro lado y en relación con los servicios web, el gran abanico de funcionalidad que dispone el ABP es tan grande que no se dispone del tiempo suficiente para desarrollar una API que comprenda todo. Existe además, una heterogeneidad en el ABP gracias a las aplicaciones y recursos en los que se apoya, la cual es una propiedad muy difícil de abarcar. Dicho esto, se debía seleccionar la funcionalidad mínima y primordial para poder considerar efectiva la solución. Se decide así centrarse en exponer la propiedad didáctica del ABP y el acceso a los datos que solo los usuarios contemplan.

La mayor dificultad encontrada en el desarrollo de los servicios web consistió en el diseño de la API y el formato de salida. Con respecto a la primera, no fue hasta muy avanzado el desarrollo del trabajo cuando se estableció finalmente las funciones y bloques que compondrían la API. Y con respecto a la segunda, las salidas generadas suponen una parte muy importante ya que no solo sirven para la obtención de información. De ellas también depende la correcta iteración entre funciones para poder acceder a todos los datos que el desarrollador puede acceder.

Finalmente, los servicios web suponen una gran ayuda no solo para la solución móvil. La API está completamente disponible para todo desarrollador que desee realizar una aplicación en comunicación con el ABP. Esto supone un logro tremendo y ayuda con creces y abre la puerta a la expansión del portal.

En conclusión, tras finalizar el desarrollo de las dos soluciones, y tras la vista de los primeros resultados y opiniones obtenidas, se puede confirmar una gran mejoría en la expansión del uso del portal. Esta expansión ha favorecido sobre todo en el acceso al ABP y en una mejora de las impresiones que los usuarios tienen de la misma.

7.2. Líneas futuras

Con todo lo realizado, todavía quedan cosas pendientes por hacer y otras nuevas surgen a raíz de este trabajo. Sin embargo, el proyecto está en sus últimos meses de desarrollo ya que finaliza en verano de 2014. Aun así, los meses restantes son más que aprovechables y se puede realizar una mayor aportación al proyecto. Existen pues varias vías de desarrollo y alternativas a partir de las soluciones desarrolladas. Algunas suponen una mejora o ampliación a las ya existentes y otras pueden surgir como nuevas ideas a incluir dentro del proyecto. A continuación se destacan las principales líneas de desarrollo, ordenadas por prioridad:

- Ampliar la funcionalidad de los servicios web: Poder permitir de manera remota funcionalidad como el registro de usuarios, edición del perfil, visualización de notificaciones, sistema de amistad para buscar amigos o solicitar peticiones de amistad, entre otros.
- Mejorar la seguridad de la App: Muchas de las peticiones pasan parámetros por URI. Las peticiones pueden ser interceptadas y los sistemas maliciosos pueden aprovechar esto para

realizar suplantaciones de identidad. Salvo que no se utilice HTTPS como protocolo de comunicación seguro, es una opción a considerar el implementar un sistema mejorado de autenticación o control de sesión.

- Ampliar la funcionalidad de la App: Acceder al resto de contenido de la comunidad de un curso (ficheros, paginas, foros de discusión, etc…), añadir sistema de mensajería del portal, visualizar perfil y crear cursos, entre otros.
- Mejorar la visibilidad de la App: Adaptarla y hacerla más parecida al ABP. Diseñar la visualización para tablets y pantallas de gran tamaño que utilicen Android.
- Diseño de la App para otras plataformas como iOS o Windows Phone. Esto ayudaría a incrementar aún más la expansión del ABP a través de dispositivos móviles.
- Diseño de nuevas aplicaciones que utilicen la API, ya sean web, aplicaciones de escritorio o móviles, siempre y cuando sea de utilidad y respete los objetivos del proyecto.
- Optimizar los algoritmos de la App y los servicios web. Todo código se puede mejorar y no es diferente para este caso. Desde reducir el tamaño de las soluciones hasta mejorar la velocidad de las tareas. Si se dispone del tiempo y los recursos necesarios, el código puede someterse a cambios y mejoras con tal de mejorar la experiencia del usuario.

8. TABLA DE ILUSTRACIONES

9. BIBLIOGRAFÍA Y REFERENCIAS

[1] Fundación Telefónica, «TIC, desarrollo y negocios inclusivos.,» Ariel, Madrid, 2012.

[2] A. Risk y J. Dzenowagis, «Review Of Internet Health Information Quality Initiatives.,» 2001.

[3] V. Gray, «África, el continente móvil,» *Economía Exterior, UIT,* nº 36, 2006.

[4] C. Juma, «El nuevo motor,» *La nueva África,* vol. Diciembre, 2011.

[5] C. Stork, E. Calandro y A. Gillwald, «Internet Going Mobile: Internet access and usage in eleven African countries,» de *Proceedings of the 19th ITS Biennial Conference 2012*, Bangkok, Tailandia, 2012.

[6] T. Brown, «M-Learning in Africa: Doing the unthinable and reaching the unreachable.,» de *International Handbook of Information Technology in Education.*, Midrand, Sudáfrica, 2007.

[7] International Telecommunication Union (ITU), «THE WORLD IN 2010,» International Telecommunication Union (ITU), Ginebra, Suiza, 2011.

[8] V. Irikefe, G. Vaidyanathan, L. Nordling, A. Twahirwa, E. Nakkazi y R. Monastersky, «Science in Africa: The view from the front line,» *Nature,* nº 474, 2011.

[9] M. L. Crescente y D. Lee, «Critical issues of m-learning: design models, adoption processes, and future trends,» *Journal of the Chinese Institute of Industrial Engineers,* nº 28, 2011.

[10] J. Donner, S. Gitau y G. Marsden, «Exploring Mobile-only Internet Use: Results of a Training Study in Urban South Africa.,» International Journal of Communication, Ciudad del Cabo, Sudáfrica, 2011.

[11] R. Calvo, A. Iglesias y L. Moreno, «A theoretical accessible approach for Collaborative Learning in mobile devices.,» de *Proceedings of 3rd International Conference on Computer Supported Education*, Noordvijkerhout, Holanda, 2011.

[12] C. Izarra, «Blog de Carolina Izarra,» Facultad de Humanidades y Educación, Universidad de Los Andes., Noviembre 2010. [En línea]. Available: http://carolinaizarra.wordpress.com/81-2/.

[13] Fundación Telefónica, «Guía de Mobile Learning,» Telefónica, 2011.

[14] D. Keegan y P. Landers, «The future of learning: From eLearning to mLearning,» Ericsson, 2002. [En línea]. Available: http://learning.ericsson.net/mlearning2/project_one/book.html.

[15] Fundación Telefónica, «Laboratorio M-Learning: Proyecto Curalia,» Telefónica, 2012. [En línea]. Available: http://laboratorios.fundaciontelefonica.com/laboratorio-ft/laboratorio-m-learning/.

[16] UNESCO, Nokia, «UNESCO Mobile Learning Week Report,» UNESCO, París, 2011.

[17] Duolingo, [En línea]. Available: https://www.duolingo.com/.

[18] Radio Televisión Española, «Clan en RTVE.es,» [En línea]. Available: http://www.rtve.es/moviles/android-clan/.

[19] Wlingua, «wlingua.com,» [En línea]. Available: http://www.wlingua.com/es/.

[20] ICWE GmbH, «Digital Learning Technologies: What, How and Why?,» *The eLearning Africa Report 2013,* 2014.

[21] T. Unwin, «UNESCO Chair in ICT for Development: Survey of e-Learning in Africa.,» UNESCO, Londres, Reino Unido, 2008.

[22] A. Jimenez Castellanos, D. de la Iglesia, M. Ramirez Robles y V. Maojo, «The AFRICA BUILD Project: Building a research and educational infrastructure for health in Africa,» Open Access Africa, Kumasi, Ghana, 2011.

[23] K. Løvborg Jensen, Remote and Autonomous Studies of Mobile and Ubiquitous Applications in Real Contexts, Aalborg, Dinamarca: IGI, 2011.

[24] S. Wasserman y K. Faust, «Social Network Analysis in the Social and Behavioral Sciences,» *Cambridge University Press,* 1994.

[25] V. Sebastián y K. F. K. Namsu Park, «Is There Social Capital in a Social Network Site? Facebook Use and College Students' Life Satisfaction, Trust, and Participation,» *Journal of Computer-Mediated Communication,* nº 14, 2009.

[26] H. Wang y B. Wellman, «Social Connectivity in America: Changes in Adult Friendship Network Size from 2002 to 2007,» *American Behavioral Scientist,* nº 53, 2010.

[27] PassMark Software, «CPU Benchmark,» PassMark Software, 2014. [En línea]. Available: http://www.cpubenchmark.net/.

[28] Android, «Android Developers,» [En línea]. Available: http://developer.android.com/develop/index.html.

[29] Apple, «Apple Developer,» [En línea]. Available: https://developer.apple.com/.

[30] Microsoft, «Windows Phone, Dev Center,» [En línea]. Available: http://devcenter.windowsphone.com/en-us.

[31] D. West, «Mobile Learning: Transforming Education and Engaging Students and Teachers.,» Washington, EEUU, 2013.

[32] W3C, «Web Services Glossary,» 2004.

[33] P. Bianco, R. Kotermanski y P. Merson, «Evaluating a Service-Oriented Architecture,» *Software Engineering Institute,* 2007.

[34] N. Bieberstein, Service-Oriented Architecture Compass, Pearson, 2006.

[35] W3C, «SOAP Specifications,» 2007.

[36] R. Fielding, «Doctoral dissertation: Representational State Transfer (REST),» UC Irvine, 2000 .

[37] Sun Microsystems, Inc., «RFC 1057 - RPC: Remote Procedure Call Protocol specification.,» 1988.

[38] A. D. Birrell y B. Jay Nelson, «Implementing Remote Procedure Calls,» ACM, 1984.

[39] V. Ranjangaonkar, «Marshalling (Serialización),» Whatis, [En línea]. Available: http://whatis.techtarget.com/definition/marshalling.

[40] Oracle, «The History of Java Technology,» [En línea]. Available: http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html.

[41] Oracle, Java Language and Virtual Machine Specifications, 8th edition, Java SE Documentation.

[42] W3C, World Wide Web Consortium, «HTML5,» 2008.

[43] PHP Group, «PHP Documentation».

[44] ISO, «ISO/IEC 9075-1:2008,» 2008.

[45] A. Beaulieu, «Learn SQL,» O'Reilly.

[46] W3C, World Wide Web Consortium, «XML,» 2003.

[47] ECMA International, «Estandar ECMA-262: ECMAScript Language Specification,» Ginebra, Suiza, 2011.

[48] ECMA International, «Introducción a JSON,» [En línea]. Available: http://www.json.org/json-es.html.

[49] Elgg, «Elgg,» 2014. [En línea]. Available: www.docs.elgg.org.

[50] Elgg, «Learn Elgg,» Elgg, [En línea]. Available: www.learn.elgg.org.

[51] Bloomberg, «Google Buys Android for Its Mobile Arsenal,» *BloombergBusinessWeek,* Agosto 2005.

[52] Google, «Android Source,» Android, 2014. [En línea]. Available: http://source.android.com/.

[53] IEEE Standar, «IEEE Recommended Practice for Software Requirements Specification,» 1998.

[54] The Apache Software Foundation, «HTTP Components,» [En línea]. Available: http://hc.apache.org/.

[55] S. Gomez Oliver, Manual de Programación Android, 2014.

[56] Google, «Android Developers,» Google, [En línea]. Available: www.developer.android.com.

[57] J. D. Sachs, «UN Millennium Project 2005. Investing in Development: A Practical Plan to Achieve the Millennium Development Goals,» United Nations Development Programme, 2005.

[58] Ericsson, «Mobile Learning: The Next Generation Of Learning.,» 2008. [En línea]. Available: http://learning.ericsson.net/mlearning2/.

10. ANEXOS

10.1. ANEXO A: Estructura de un Plugin en Elgg

Aquí se define la estructura del plugin para la implementación de los servicios. Un plugin de Elgg contiene en su máximo despliegue una completa estructura de carpetas y ficheros para así poder acceder a cualquier parte del sistema que comprende el engine; sin embargo, solo se va a describir el mínimo necesario para una correcta implementación de los servicios web [49].

A continuación se muestra el desglose de un plugin "ejemplo" para la versión 1.8 de Elgg. Todos los plugins deben ser desplegados en la carpeta "mod" del sistema: `/mod/example/`.

```
actions/
    example/
        action.php
        other_action.php
languages/
    en.php
lib/
    example.php
pages/
    example/
        all.php
        owner.php
views/
    default/
        example/
            css.php
        forms/
            example/
                action.php
                other_action.php
COPYRIGHT.txt
INSTALL.txt
manifest.xml
README.txt

start.php
```

Fig. 44 Estructura de un plugin para Elgg 1.8

Todos los plugins **deben contener como mínimo los ficheros start.php y el manifest.xml en el directorio raíz del plugin** con el fin de ser reconocidos por Elgg.

10.1.1.1. Start.php

Cada plugin debe contener este fichero, aunque esté vacío, ya que es llamado por el engine de Elgg solo para activar el plugin dentro del portal. En este fichero es donde el plugin carga y registra el código entre otros. También suele tener los siguientes usos:

- Registro de los eventos y las funciones a utilizar.
- Registro de acciones
- Registros de páginas
- Extensión de vistas
- Registros de CSS o JS

- Añadido de entradas en menús
- Incluir y/o referenciar las librerías adicionales necesarias para el funcionamiento del plugin.

10.1.1.2. Ficheros .txt

El plugin puede contener los siguientes ficheros de textos, cada uno con información específica. Si están bien formados, Elgg los reconoce y los enlaza para poder ser accesibles desde la interfaz:

- **README.txt**: Proporciona información adicional sobre el plugin.
- **COPYRIGHT.txt**: Si se incluye, debe proporcionar una definición de los derechos de autor del plugin.
- **INSTALL.txt**: Proporciona instrucciones adicionales para instalar el plugin en el caso de que el proceso fuera más complejo que la instalación de plugins para Elgg por defecto.

10.1.1.3. Carpeta /lib/

Consiste en una carpeta donde el engine de Elgg almacena su código asociado a procedimientos auxiliares a modo de librerías. No hay obligación específica de introducir funcionalidad en esta carpeta, sin embargo, es recomendable el uso del formato para fomentar la coherencia y la identificación de los recursos para cualquier plugin de Elgg.

10.1.2. Construcción de una API HTTP para Elgg

Elgg proporciona un framework completo y necesario para la construcción de servicios web. Dicho framework permite exponer la funcionalidad fuera del sitio y sigue la arquitectura REST/RPC híbrida, similar a las APIs proporcionadas por sitios como Flickr o Twitter [50].

Para crear la API para el portal se necesitan aplicar tres conceptos:

- Exponer métodos
- Configurar la autenticación de la API
- Configurar la autenticación de usuario.

10.1.2.1. Exponer métodos

La función usada en Elgg para exponer métodos es "*expose_function()*". Dicha función recibe los siguientes parámetros:

- El nombre del método externo con el que será llamado desde fuera.
- El nombre de la función a exponer.
- Los parámetros de entrada de la función.
- Descripción breve.
- Tipo de función HTTP (GET o POST).
- Si requiere autenticación API (verdadero o falso).
- Si requiere autenticación de usuario (…).

A continuación se describe un ejemplo de función a exponer:

```
function my_echo($string) {
    return $string;
}


expose_function("test.echo",
                "my_echo",
                 array("string" => array('type' => 'string')),
                 'A testing method which echos back a string',
                 'GET',
                 false,
                 false
                );
```

Fig. 45 Ejemplo de Función expuesta en Elgg

Una vez introducido y compilado el código, la función es accesible a través de la siguiente URL:

http://yoursite.com/services/api/rest/xml/?method=test.echo&string=testing

En dicha URL podemos ver los parámetros correspondientes al nombre de la función y sus parámetros asociados. La cabecera *XML/* es utilizado para devolver la información en formato XML; pero puede usarse la cabecera *JSON/* para devolver su equivalente en JSON.

10.1.2.2. Autenticación de la API

En el caso de que se desea controlar el acceso a las funciones expuestas, la Autenticación API permite controlar el acceso a los mismos, por ejemplo, cuando se necesitan dichas funciones en otra plataforma dentro del server, limitar la cantidad de desarrolladores externos que tienen acceso a tu API o cuando deseas limitar el número de llamadas de un desarrollador a dicha API al día [50].

El sistema de autenticación consiste en proporcionar una clave HMAC al usuario, de manera similar a como lo hacen Google, Flickr o Twitter. Dicha clave permite solicitar un token al desarrollador ya con los permisos necesarios para poder acceder a la API [50].

Para nuestro caso, omitiremos dicho control ya que las funciones a exponer no son críticas y el proyecto Africa Build no desea gastar recursos en dicho control.

10.1.2.3. Autenticación de usuario

Este sistema de autenticación consiste en un sistema de Tokens que permiten identificar a usuarios validados dentro del sistema. El objetivo de dicho control es evitar la edición añadido o acceso al contenido del portal para usuarios no registrados en él. Para ello, antes de realizar ninguna llamada a la API, se debe hacer una llamada preliminar a la función externa "*auth.gettoken*". Dicha función

solicita el nombre de usuario y contraseña del portal. Si la petición es correcta, la función devuelve un token identificativo del usuario [50].

A partir de entonces, todas las llamadas a la API deben ir acompañadas del parámetro *auth_token*. Dicho parámetro no solo sirve de identificación; permite tener una sesión dentro del sistema para el control de los permisos del usuario y conseguir información relacionada con el mismo sin la necesidad de mandar otros parámetros identificativos en consultas posteriores (nombre, guid del usuario, remitente email, etc…). Por motivos de seguridad implementados en Elgg por defecto, el token es temporal y tiene una duración de una hora. Una vez caducado, se debe solicitar uno nuevo para poder seguir accediendo al sistema.

10.1.2.4. System.api.list

Función expuesta por defecto junto con *auth.gettoken*, proporciona una documentación breve en formato XML o JSON de las funciones expuestas en el portal. Dicha documentación contiene una descripción de cada método, su función asociada, su tipo de petición (GET o POST), parámetros de entrada y valores por defecto. La URL por defecto y la respuesta en formato JSON es de la forma que se presenta a continuación:

http://yoursite.com/services/api/rest/json/?method=system.api.list

```
{
status: 0,
result:
{
        auth.gettoken:
        {
                description: "Esta llamada a la API permite al usuario obtener
                el token de autenticación el cual puede ser utilizado para
                autenticar futuras llamadas a la API. Pásalo como el parámetro
                auth_token",
                function: "auth_gettoken",
                parameters:
                {
                        username:
                        {
                                type: "string",
                                required: true
                        },
                        password:
                        {
                                type: "string",
                                required: true
                        }
                },
                call_method: "POST",
                require_api_auth: false,
                require_user_auth: false
        },
```

```
            system.api.list:
            {
                  description: "Lista todas las llamadas a la API disponibles en
                  el sistema.",
                  function: "list_all_apis",
                  call_method: "GET",
                  require_api_auth: false,
                  require_user_auth: false
            }
}
}
```

Fig. 46 Ejemplo de respuesta de llamada a System.api.list en JSON

10.1.2.5. Formato de respuesta

Elgg proporciona una estructura básica de formato de respuesta a las peticiones tanto en JSON como XML. Dicho formato tiene como cabecera principal la etiqueta *elgg* con los campos *status* y *result*. El primero proporciona información sobre si la petición se ha realizado satisfactoriamente, indicándose así con el valor 0. El segundo contiene la respuesta de la consulta cuya estructura se define por el desarrollador de la API.

```
▼<elgg>
   <status type="integer">0</status>
   <result type="string">Hi . I'm working</result>
 </elgg>
```

Fig. 47 Ejemplo de respuesta XML en Elgg

10.2. ANEXO B: Estructura de un proyecto en Android

Al igual que ocurre con el plugin de los servicios web, en Android las Apps existentes siguen un patrón de diseño y un sistema de ficheros común. En la siguiente imagen vemos los elementos creados inicialmente para un nuevo proyecto Android [55]:

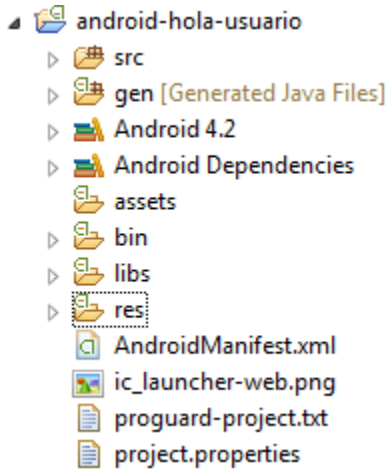


Fig. 48 Sistema de Ficheros de una App de Android

En los siguientes apartados describiremos los elementos principales de esta estructura.

10.2.1.1. Carpeta /src/

Esta carpeta contendrá todo el código fuente de la aplicación, código de la interfaz gráfica, clases auxiliares, etc... siempre bajo la estructura del paquete java definido.

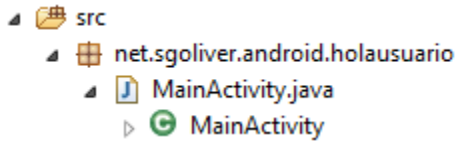


Fig. 49 Carpeta /src/

10.2.1.2. Carpeta /res/

Contiene todos los ficheros de recursos necesarios para el proyecto: imágenes, vídeos, cadenas de texto, etc. Los diferentes tipos de recursos se distribuyen entre las siguientes subcarpetas:

Carpeta	Descripción
/res/drawable/	Contiene las imágenes [y otros elementos gráficos] usados en por la aplicación. Para definir diferentes recursos dependiendo de la resolución y densidad de la pantalla del dispositivo se suele dividir en varias subcarpetas: • /drawable-ldpi (densidad baja) • /drawable-mdpi (densidad media) • /drawable-hdpi (densidad alta) • /drawable-xhdpi (densidad muy alta)

/res/layout/	Contiene los ficheros de definición XML de las diferentes pantallas de la interfaz gráfica. Para definir distintos *layouts* dependiendo de la orientación del dispositivo se puede dividir en dos subcarpetas: • /layout (vertical) • /layout-land (horizontal)
/res/anim/ /res/animator/	Contienen la definición de las animaciones utilizadas por la aplicación.
/res/color/	Contiene ficheros XML de definición de colores según estado.
/res/menu/	Contiene la definición XML de los menús de la aplicación.
/res/xml/	Contiene otros ficheros XML de datos utilizados por la aplicación.
/res/raw/	Contiene recursos adicionales, normalmente en formato distinto a XML, que no se incluyan en el resto de carpetas de recursos.
/res/values/	Contiene otros ficheros XML de recursos de la aplicación, como por ejemplo cadenas de texto (*strings.xml*), estilos (*styles.xml*), colores (*colors.xml*), arrays de valores (*arrays.xml*), etc.

No todas estas carpetas tienen por qué aparecer en cada proyecto Android, tan sólo las que se necesiten. Como ejemplo, para un proyecto nuevo Android, se crean por defecto los siguientes recursos para la aplicación:

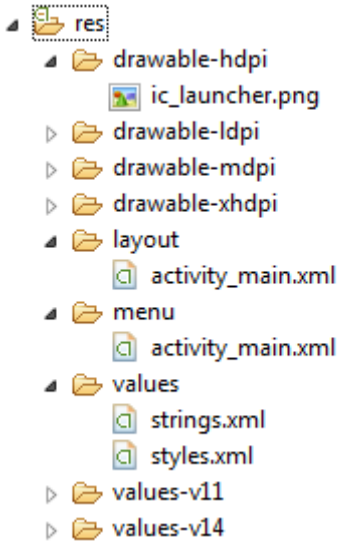


Fig. 50 Carpeta /res/

Como se puede observar, existen algunas carpetas en cuyo nombre se incluye un sufijo adicional, como por ejemplo “values-v11” y “values-v14”. Estos, y otros sufijos, se emplean para definir recursos independientes para determinados dispositivos según sus características. De esta forma, por ejemplo, los recursos incluidos en la carpeta “values-v11” se aplicarían tan sólo a dispositivos cuya versión de Android sea la 3.0 (API 11) o superior. Al igual que el sufijo “–v” existen otros muchos para referirse a otras características del terminal, puede consultarse la lista completa en la documentación oficial de Android [56].

10.2.1.3. Carpeta /gen/

Contiene una serie de elementos de código **generados automáticamente** al compilar el proyecto. Cada vez que generamos nuestro proyecto, la maquinaria de compilación de Android genera por nosotros una serie de ficheros fuente java dirigidos al control de los recursos de la aplicación. Dado que estos ficheros se generan automáticamente tras cada compilación del proyecto es importante que no se modifiquen manualmente bajo ninguna circunstancia.

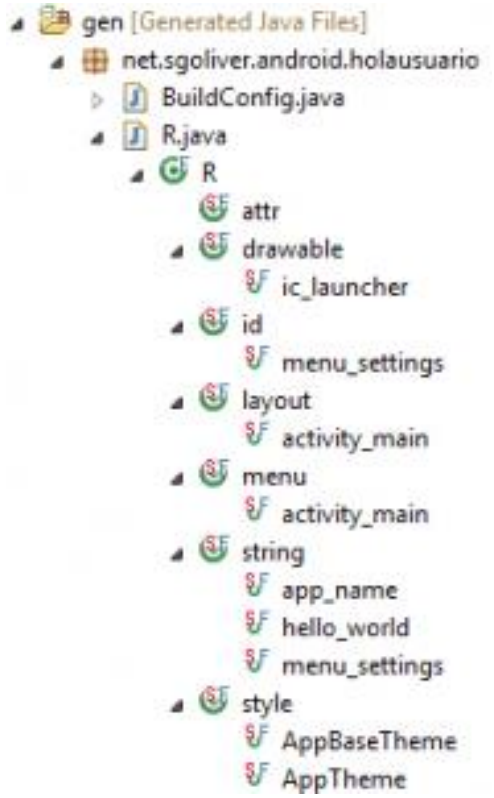


Fig. 51 Carpeta /gen/

La clase R contendrá en todo momento una serie de constantes con los ID de todos los recursos de la aplicación incluidos en la carpeta /res/, de forma que podamos acceder fácilmente a estos recursos desde nuestro código a través de este dato. Consiste así en una simple lista de constantes y punteros a posiciones de memoria.

10.2.1.4. Carpeta /assets/

Contiene todos los demás ficheros auxiliares necesarios para la aplicación (y que se incluirán en su propio paquete), como por ejemplo ficheros de configuración, datos, etc…

La diferencia entre los recursos incluidos en la carpeta /res/raw/ y los incluidos en la carpeta/assets/ es que para los primeros se generará un ID en la clase R y se deberá acceder a ellos con los diferentes métodos de acceso a recursos. Para los segundos sin embargo no se generarán ID y se podrá acceder a ellos por su ruta como a cualquier otro fichero del sistema.

10.2.1.5. Carpeta /bin/

Ésta es otra de ésas carpetas que en principio no podemos tocar. Contiene los elementos compilados de la aplicación y otros ficheros auxiliares. Cabe destacar el fichero con extensión ".apk", que es el ejecutable de la aplicación que se instalará en el dispositivo.

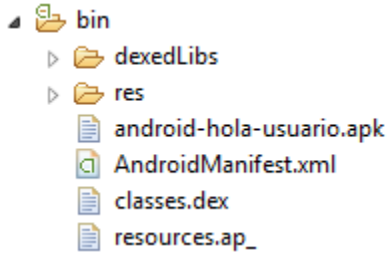


Fig. 52 Carpeta /bin/

10.2.1.6. Carpeta /libs/

Contiene las librerías auxiliares, normalmente en formato ".jar" que utilicemos en la aplicación Android.

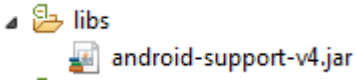


Fig. 53 Carpeta /libs/

10.2.1.7. Fichero AndroidManifest.xml

Contiene la definición en XML de los aspectos principales de la aplicación, como por ejemplo su identificación (nombre, versión, icono, etc…), sus componentes (pantallas, mensajes, etc…), las librerías auxiliares utilizadas, o los permisos necesarios para su ejecución.

10.2.2. Componentes

A continuación se describen los componentes utilizados para el desarrollo de la App [55].

10.2.2.1. Activity

Las actividades representan el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana o pantalla en cualquier otro lenguaje visual.

10.2.2.2. View

Las vistas son los componentes básicos con los que se construye la interfaz gráfica de la aplicación, análoga por ejemplo a los controles de Java. De inicio, Android pone a nuestra disposición una gran cantidad de controles básicos, como cuadros de texto, botones, listas desplegables o imágenes, aunque también existe la posibilidad de extender la funcionalidad de estos controles básicos o crear nuestros propios controles personalizados.

10.2.2.3. Service

Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son similares a los servicios presentes en cualquier otro sistema operativo. Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (p.ej. actividades) si se necesita en algún momento la interacción con del usuario.

10.2.2.4. Intent

Un intent es el elemento básico de comunicación entre los distintos componentes Android que hemos descrito anteriormente. Se pueden entender como los mensajes o peticiones que son enviados entre los distintos componentes de una aplicación o entre distintas aplicaciones. Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc…

10.3. ANEXO C: Documentación de la API de Servicios Web (Inglés)

10.3.1. Introduction

The present document shows the Web Services Layer in high-level for the ABP. The purpose of this layer is to provide access to portal resources through external applications or systems. The API, currently under development, uses REST architecture and it will become part of a plugin for use. Here are listed the expose functions for use, organized into the following sections.

For the correct use of the API, the URI of access to the resources is next:

http://egasmoniz.dia.fi.upm.es/elgg/services/api/rest/***{xml|json}***/
?method=***{function}[&param=value]***

All responses return a status tag indicating whether the request was successful (value 0, -1 if an error occurs), along with the result tag with the content.

10.3.2. Auxiliar Functions

REST & Param. Functions	Parameters	Description	Output
GET abp_aux_hello	user_token	Returns a simple request for testing the communication with the portal.	Hello World text.
GET abp_aux_site	user_token	Returns information about the portal.	ABP and Moodle info.
GET system.api.list	-	Returns all API functions available.	A list of API functions each with their input and output parameters
GET auth.gettoken	username password	Check if the user is registered in the portal given his username or email and his pass. The token has a duration of 1 hour per user session.	User token
GET abp_aux_user	username = null user_token	Get user profile information given his username or the logged username if null.	Returns user's profile

10.3.2.1. ABP_AUX_HELLO

- Method: GET
- Parameters: auth.gettoken (String).
- Schema:

```
{
    "status": 0,
    "result": "Hi [user logged in]. I'm working"
```

```
}
```

10.3.2.2. ABP_AUX_SITE

- Method: GET
- Parameters: auth.gettoken (String).
- Schema:

```
{
    "status": 0,
    "result": {
        "url": "[URL del sitio]",
        "sitename": "[Site name]",
        "language": "[language code]"
    }
}
```

10.3.2.3. SYSTEM.API.LIST

- Method: GET
- Parameters: -
- Schema:

```
{
    "status": 0,
    "result": {
        "[Function name]": {
            "description": "[description]",
            "function": "plugin function name",
            "call_method": "[GET | POST]",
            "require_api_auth": [true | false],
            "require_user_auth": [true | false]
        },
       "[Function name 2]": {
          …
     }
}
```

10.3.2.4. AUTH.GETTOKEN

- Method: POST
- Parameters: username (String), password (String).
- Schema:

```
{
    "status": 0,
    "result": "[token value]"
}
```

10.3.2.5. ABP_AUX_USER

- Method: GET
- Parameters: auth.gettoken (String).
- Schema:

```
{
    "status": 0,
    "result": {
```

```
        "core": {
            "name": "[name]",
            "username": "[username]"
        },
        "profile_fields": {
            "[fieldname 1]": {
                "label": "[label name]",
                "type": "[label type]",
                "value": "[value]"
            },
            "[fieldname 2]": {
                …
            },
        …
        },
        "avatar_url": "[URL of user avatar]"
    }
}
```

10.3.3. Entities

REST & Param. Functions	Parameters	Description	Output
GET abp_entity_get	id_entity user_token	Returns complete information of an entity object given an Id (Course, Section, User, Site, Quiz, Url, Blog, Bookmark, Message, Notification, Discussion and File).	The entity with metadata fields.
POST abp_entity_comment	id_entity text user_token	Send a comment of a user to an entity.	OK value
POST abp_entity_enroll	Id_course user_token	Send the value of user enrollment course.	Enroll value
POST abp_entity_like	Id_entity user_token	Send the like value of an entity.	Like value
POST abp_entity_unlike	Id_entity user_token	Delete a like.	Like value
GET abp_entity_likes_count	Id_entity user_token	Count number of likes of an entity.	Like count
GET abp_entity_likes_users	Id_entity user_token	Get users who liked an entity.	List of users

10.3.3.1. ABP_ENTITY_GET

- Method: GET
- Parameters: id_entity (Integer), user_token (String)
- Schema:

```
{
```

```
    "status": 0,
    "result": {
        "guid": [guid],
        "type": [type],
        "title": [title],
        "subtype": [subtype],
        "container_guid": [container_guid],
        "description": “[description]”,
        "owner_guid": [owner_guid],
        "time_created": "[n] days ago",
        "icon": "[URL icon]",
        "metadata": [{
            "metadata_guid": "[metadata_guid]",
            "metadata_name": "[metadata_name]",
            "metadata_value": "[metadata_value]",
            "metadata_time_created": "[m] days ago"
        }, {
          …
        }]
    }
}
```

10.3.3.2. ABP_ENTITY_COMMENT

- Method: POST
- Parameters: id_entity (Integer), text (String)
- Schema:

```
{
    "status": 0,
    "result": {
        "success": {
            "message": "Your comment was successfully posted."
        }
    }
}
```

10.3.3.3. ABP_ENTITY_GET

- Method: POST
- Parameters: id_course (Integer)
- Schema:

```
{
    "status": 0,
    "result": "[unenroll | enroll] successfully"
}
```

10.3.3.4. ABP_ENTITY_LIKE

- Method: POST
- Parameters: id_entity (Integer)
- Schema:

```
{
    "status": 0,
    "result": "You now like this item"
}
```

10.3.3.5. ABP_ENTITY_UNLIKE

- Method: POST
- Parameters: id_entity (Integer)
- Schema:

```
{
    "status": 0,
    "result": "Your like has been removed"
}
```

10.3.3.6. ABP_ENTITY_LIKES_COUNT

- Method: GET
- Parameters: id_entity (Integer)
- Schema:

```
{
    "status": 0,
    "result": "[n]"
}
```

10.3.3.7. ABP_ENTITY_LIKES_USERS

- Method: GET
- Parameters: id_entity (Integer)
- Schema:

```
{
    "status": 0,
    "result": [{
        "id": "[id_annotation]",
        "owner_guid": "[id_user]",
        "name": "[username]",
        "time_created": "[n] days ago"
    }, {
        …
    }]
}
```

10.3.4. Learning Resources

REST & Param. Functions	Parameters	Description	Output
GET abp_learning_get	id_resource user_token	Returns complete data of an e-learning resource given an GUID (Video, DUDAL, File and Quiz)	Complete quiz data
POST abp_learning_post_answers	Id_quiz Id_unique	Send the user answers of a quiz given the id of the quiz and the id that associates the user with the quiz for evaluation in this attempt. Returns solutions quiz and user grade	Solutions Quiz and user grade

10.3.4.1. ABP_LEARNING_GET

- Method: GET
- Parameters: id_resource (String), user_token (String).
- Example:


```
{
    "status": 0,
    "result": {
        "resource_type": "quiz",
        "questionlist": "49,truefalse.50,truefalse",
        "content": {
            "quizid": "14",
            "userid": "9123",
            "course": "5",
            "name": "Questionnaire 1 (1 min)",
            "intro": "Description for the Questionnaire 1 bis",
            "timeopen": "0",
            "timeclose": "0",
            "preferredbehaviour": "deferredfeedback",
            "grademethod": "1",
            "shufflequestions": "0",
            "shuffleanswers": "1",
            "questions": "49,50,0,0",
            "layout": "1,2",
            "sumgrades": "2.00000",
            "gradequiz": "100.00000",
            "timelimit": "60",
            "attempts": "0",
            "attemptsuser": "43",
            "attemptonlast": "0",
            "timemodified": "1357737509",
            "enable": "1",
            "scorebelow": "",
            "grade_user": {
                "grade": "100.00000",
                "timemodified": "1403269964"
            },
```

```
            "questionlist": [{
                "id": "49",
                "slot": "1",
                "name": "Pregunta 1",
                "questiontext": "\u00bfMuri\u00f3 la madre de Bambi en la
pel\u00edcula?",
                "qtype": "truefalse",
                "penalty": "1.0000000",
                "grade": "1.0000000",
                "variables": [],
                "answers": [{
                    "idanswer": "167",
                    "answer": "False",
                    "questiontext": "",
                    "answertext": "",
                    "answertype": ""
                }, {
                    "idanswer": "166",
                    "answer": "True",
                    "questiontext": "",
                    "answertext": "",
                    "answertype": ""
                }],
                "numerical_options": {
                    "array_units": [],
                    "unit_type": "",
                    "unit_list": "",
                    "unit_place": "",
                    "unit_penalty": ""
                },
                "essay_options": "",
                "single": ""
            }, {
                "id": "50",
                "slot": "2",
                "name": "Pregunta 2",
                "questiontext": "\u00bfEs cierto que en Ben Hur- algunos de los
\"prisoneros\" que reman en la galera romana, llevan puestos relojes de pulsera?",
                "qtype": "truefalse",
                "penalty": "1.0000000",
                "grade": "1.0000000",
                "variables": [],
                "answers": [{
                    "idanswer": "169",
                    "answer": "Falso",
                    "questiontext": "",
                    "answertext": "",
                    "answertype": ""
                }, {
                    "idanswer": "168",
                    "answer": "Verdadero",
                    "questiontext": "",
                    "answertext": "",
                    "answertype": ""
                }],
                "numerical_options": {
```

```
                    "array_units": [],
                    "unit_type": "",
                    "unit_list": "",
                    "unit_place": "",
                    "unit_penalty": ""
                },
                "essay_options": "",
                "single": ""
            }],
            "uniqueid": "920",
            "timestart": "1403802663"
        }
    }
}
```


10.3.4.2. ABP_LEARNING_POST_ANSWERS

- Method: POST
- Parameters: id_quiz (Integer), id_unique (Integer), user_token (String).
- Example:


```
{
"status": 0,
"result": {
"quizid": "13",
"maxgradequiz": "100.00000",
"quizusergrade": "20",
"generalusergrade": "20.00000",
"automaticgrade": "1",
"totaloutstandingquestions": "0",
"feedbackquiz": "\n\nMuy buen trabajo",
"infoquestion": [
{
        "id": "57",
        "questiontext": "Instrucciones.",
        "type": "description",
        "questiongrade": "0",
        "questionusergrade": "0",
        "rightanswer": "",
        "generalfeedback": "",
        "combinedfeedback": "",
        "numcorrect": ""
},
{
        "id": "54",
        "questiontext": "\n\n¿Cuánto es 1+6.4?",
        "type": "calculatedsimple",
        "questiongrade": "10",
        "questionusergrade": "10",
        "rightanswer": "7.4 ",
        "generalfeedback": "",
        "feedbackanswer": [
                {
                "answer": "7.4 ",
                "feedback": "",
```

```
                "correct": ""
                }
        ],
        "combinedfeedback": "",
        "numcorrect": ""
},
{
        "id": "51",
        "questiontext": "\n\n¿Cuánto son 3+2?",
        "type": "numerical",
        "questiongrade": "10",
        "questionusergrade": "10",
        "rightanswer": "5",
        "generalfeedback": "",
        "feedbackanswer": [
                {
                "answer": "5 ",
                "feedback": "Muy bien",
                "correct": ""
                }
        ],
        "combinedfeedback": "",
        "numcorrect": ""
},
{
        "id": "58",
        "questiontext": "\n\n¿Cuál de la respuesta correcta?\n- La Tierra es un
        planeta.\n- El Sol es un planeta.\n- La Luna es una planeta.",
        "type": "shortanswer",
        "questiongrade": "10",
        "questionusergrade": "0",
        "rightanswer": "La Tierra es un planeta",
        "generalfeedback": "",
        "feedbackanswer": [
                {
                "answer": "",
                "feedback": "",
                "correct": ""
                }
        ],
        "combinedfeedback": "",
        "numcorrect": ""
},
{
        "id": "61",
        "questiontext": "\n\n¿Cuánto son 3+ 5.5?: 6.50; 10.50; 8.50",
        "type": "calculatedmulti",
        "questiongrade": "10",
        "questionusergrade": "0",
        "rightanswer": "8.50",
        "generalfeedback": "",
        "feedbackanswer": [
                {
                "answer": "",
                "feedback": "",
                "correct": ""
```

```
                }
        ],
        "combinedfeedback": "",
        "numcorrect": ""
},
{
        "id": "52",
        "questiontext": "\n\nUne correctamente: {\n\nSiglas de la Universidad
        Politécnica de Madrid;\n\nNúmero de estaciones del año;\n\nCapital de Madrid}
        -> {Madrid; Cuatro; UPM}",
        "type": "match",
        "questiongrade": "10",
        "questionusergrade": "0",
        "rightanswer": "\n\nSiglas de la Universidad Politécnica de Madrid -
        UPM;\n\nNúmero de estaciones del año - Cuatro;\n\nCapital de Madrid - Madrid",
        "generalfeedback": "",
        "feedbackanswer": [
                {
                "answer": "Siglas de la Universidad Politécnica de Madrid - ",
                "feedback": "",
                "correct": "0"
                },
                {
                "answer": "Número de estaciones del año - ",
                "feedback": "",
                "correct": "0"
                },
                {
                "answer": "Capital de Madrid -> ",
                "feedback": "",
                "correct": "-1"
                }
        ],
        "combinedfeedback": "",
        "numcorrect": ""
},
{
        "id": "55",
        "questiontext": "\n\n¿Ciudades de
        España?:\n\nBogotá;\n\nMadrid;\n\nAlbacete;\n\nParís",
        "type": "multichoice",
        "questiongrade": "10",
        "questionusergrade": "0",
        "rightanswer": "\n\nMadrid",
        "generalfeedback": "",
        "feedbackanswer": [
                {
                "answer": "",
                "feedback": "",
                "correct": ""
                }
        ],
        "combinedfeedback": "",
        "numcorrect": ""
},
{
```

```
        "id": "53",
        "questiontext": "\n\n¿Cuántos kilos son 6.3 libra 6.0?",
        "type": "calculated",
        "questiongrade": "10",
        "questionusergrade": "0",
        "rightanswer": "13.889133847158 ",
        "generalfeedback": "",
        "feedbackanswer": [
                {
                "answer": " ",
                "feedback": "",
                "correct": ""
                }
        ],
        "combinedfeedback": "",
        "numcorrect": ""
}
]
}
```


10.3.5. Lists

REST & Param. Functions	Parameters	Description	Output
GET abp_list_courses	flag_course = all limit = 10 offset = 0 user_token	Returns the available courses for a user given a flag (all, colleagues or mine).	A short list of courses filtered by the flag.
GET abp_list_users	id_entity = null limit = 10 offset = 0 user_token	Returns a list of users with a relationship with an entity given your id (or null for all users). In other case the list is for users who marked that value in the entity.	A short list of users
GET abp_list_entity	type = object subtype = null container_guid = null limit = 10 offset = 0 user_token	Returns a list of entities. The list can be filtered by the flag type (Object, User, Group or Site), subtype (Courses, blogs, bookmarks…) and container (blogs, bookmarks, sites… from another entity).	A short list of courses.
GET abp_list_activity	Id_entity = null limit = 10 offset = 0 user_token	Returns the activity. The id can correspond to an entity or the main activity if it is null.	A short list of activity filtered.

10.3.5.1. ABP_LIST_COURSES

- Method: GET
- Parameters: flag_course (String), limit (Integer), offset (Integer), user_token (String).
- Schema:

```
{
    "status": 0,
    "result": [{
        "guid": [guid_course],
        "title": "[title]",
        "description": "[description]",
        "startdate": "[n] days ago",
        "icon": "[URL icon]"
    }, {
     …
    }]
}
```

10.3.5.2. ABP_LIST_USERS

- Method: GET
- Parameters: id_entity (Integer), limit (Integer), offset (Integer), user_token (String).
- Schema:

```
{
    "status": 0,
    "result": [{
        "guid": [guid_user],
        "name": "[name]",
        "username": "[username]",
        "email": "[email]",
        "avatar_url": "[URL avatar]"
    }, {
    …
    }]
}
```

10.3.5.3. ABP_LIST_ENTITY

- Method: GET
- Parameters: type (String), subtype (String), id_container (Integer), limit (Integer), offset (Integer), user_token (String).
- Schema:

```
{
    "status": 0,
    "result": [{
        "guid": [guid],
        "title": "[title]",
        "subtype": "[subtype]",
        "container_guid": [entity container guid],
```

```
        "created": "[day]",
        "owner_guid": "[owner guid]",
        "owner_name": "[owner username]",
        "icon": "[URL icon]"
    }, {
    ---
    }]
}
```

10.3.5.4. ABP_LIST_ACTIVITY

- Method: GET
- Parameters: id_entity (Integer), limit (Integer), offset (Integer), user_token (String).
- Schema:

```
{
    "status": 0,
    "result": [{
        "subject_guid": subject_guid,
        "subject_name": "[subject_name]",
        "object_guid": [object_guid],
        "object_title": "[object_title]",
        "type": "[type]",
        "subtype": "[subtype]",
        "action_type": "[type of action]",
        "posted": "[date]"
    }, {
    …
    }]
}
```

10.3.6. Messages

REST & Param. Functions	Parameters	Description	Output
GET abp_message_count	user_token	Get a count of the unread messages.	Count value
GET abp_message_inbox	limit = 10 offset = 0 user_token	Get messages inbox.	Short list of received messages
GET abp_message_read	Id_message user_token	Read a single message.	Message's header and his content
POST abp_message_sent	limit = 10 offset = 0 user_token	Get sent messages.	Short list of sent messages
POST abp_message_send	Subject Body send_to from user_token	Send a message.	Sent value

10.3.6.1. ABP_MESSAGE_COUNT

- Method: GET
- Parameters: user_token (String).
- Schema:

```
{
    "status": 0,
    "result": [n]
}
```

10.3.6.2. ABP_MESSAGE_INBOX

- Method: GET
- Parameters: limit (Integer), offset (Integer), user_token (String).
- Schema:

```
{
    "status": 0,
    "result": [{
        "guid": [id_message],
        "subject": "[subject]",
        "user": {
            "guid": [guid],
            "name": "[name]",
            "username": "[username]",
            "avatar_url": "[URL icon]"
        },
```

```
        "timestamp": [timestamp],
        "description": "[email content]",
        "read": " [yes | no]"
    }, {
     …
    }]
}
```

10.3.6.3. ABP_MESSAGE_READ

- Method: GET
- Parameters: id_message (Integer), user_token (String).
- Schema:

```
{
    "status": 0,
    "result": {
        "guid": [id_message],
        "[id_message]": {
            "subject": "[subject]"
        },
        "user": {
            "guid": [guid user],
            "name": "[name]",
            "username": "[username]",
            "avatar_url": "[URL icon]"
        },
        "timestamp": "[timestamp]",
        "description": "[content]",
        "read": "yes"
    }
}
```

10.3.6.4. ABP_MESSAGE_SENT

- Method: GET
- Parameters: limit (Integer), offset (Integer), user_token (String).
- Schema:

```
{
    "status": 0,
    "result": [{
        "guid": [id_message],
        "subject": "[subject]",
        "user": {
            "guid": [guid],
            "name": "[name]",
            "username": "[username]",
            "avatar_url": "[URL icon]"
        },
        "timestamp": [timestamp],
        "description": "[email content]",
        "read": " [yes | no]"
```

```
    }, {
     …
    }]
}
```

10.3.6.5. ABP_MESSAGE_SEND

- Method: POST
- Parameters: Subject (String), Body (String), send_to (String), from (String), user_token (String)
- Schema:

```
{
    "status": 0,
    "result": [guid_message send]
}
```